(英文タイトル)

Computer-Assisted Animation Creation Techniques for Hair Animation and Shade, Highlight, and Shadow

(和文タイトル)

アニメーション制作工程における頭髪及び陰影表現の実現

2010 年 02 月

早稲田大学大学院 理工学研究科

物理学及び応用物理学専攻 画像情報処理研究

杉崎 英嗣

Abstract

Japanese style of cartoon animation so-called anime has been attracting worldwide attention with their artistic picture quality and unique story lines. To keep the quality and the feature, there are still a lot of manual labors remaining such as hand-drawing process in the production pipeline, although using 3DCG technique has become common in the pipeline to reduce the labor intensity. In this paper, therefore, two main issues, hair expressions, and shade, highlight, and shadow expressions, are basically described.

As for hair expressions, two approaches are introduced to achieve hair animations with anime-like feature.

One is for extracting hair motion feature from the existing animation sequence. This demonstrates how to create cartoon hair animation accentuated in anime-like motions. The novelty of this approach is that users can extract the motion from the existing animation such as windy scene or motion of grass and fire etc, and then they can apply the extracted motion to other characters or objects having a three-dimensional structure. In fact, users can re-use an existing animation sequence as an input for animating 3D characters as if both existing 2D character in the input animation and 3D characters had existed from the beginning in the same environment. In addition, even if users are not an animator, to create abundant hair motion database from those existing animations with various animators' characteristic enables users to achieve hair motion with users' favorite styles characteristic.

The other is to create hair animation based on key frames. In-between creation is one of the most time-consuming and labor-intensive procedures in actual production especially hair motion in Anime. Much research has been applied in automatic in-between generation but it is still a challenging task. Therefore, proposed method is to create in-between for hair animation aiming at the actual animation production. This technique reduces time needed for the production by creating automatic in-betweening based on our simulation model. The inputs required are several key frames for the character's hair drawn by animators. First, the skeleton structure on the hair strands is automatically constructed by a contour-based extraction method. Next, for the several situations configured by animators such as windblown hair or swinging hair with the motion of the character's head, motions of the skeleton model are created based on the forward dynamics algorithm. This approach can create the hair motion to converge onto

the target hair strands by applying a force called “Convergence Force” to the hair motion. And then, a shape of the hair strands is deformed by the results of motion synthesis of its skeleton. To avoid unnatural shapes such as a self-intersection during the deformation, a shape preservation method is applied to the hair strands. In-between frames themselves are automatically generated by using the simulation model in this way. Because the simulation continues until the hair motion converges onto the target hair strands, the obtained in-between frames are more or less accordingly to the animator’s design. This approach provides a feature frame selection algorithm for animators after obtaining the in-between frames. According to the animator’s specification, a certain number of in-between frames are selected automatically. Comparing to previous models, this simulation model can faithfully adhere to the input key frames, even though a simulation-based motion model is utilized. Animator-oriented hair animation is thus achieved by applying our method to each key frame.

In terms of shade and highlight, and shadow, two of the pipelines have been developed. One is for shading and highlighting, and the other is shadowing. Since the shadow is rendered after the characters or objects layers are drawn in the real production pipeline, these two processes are naturally separated.

A semi-automatic shading pipeline for 2D animation is proposed, especially for Japanese anime style, in this paper. In traditional 2D animation production, each stroke in every frames, including the highlight and shade strokes, has been drawn by hand. The proposed technique reduces such time needed for the production by automatic shade and highlight generation based on a vector inpainting method. The only input required is a hand-drawn character or object with strokes. First, initial vectors are distributed on the strokes according to several conditions of the stroke relationship such as open, closed, and connected or animator’s preference. From these initial vectors, the pipeline proposes an energy minimization with the divergence free constraints to inpaint vectors for the whole character or object region. Then, based on the inpainted vectors, the toon-shading technique is applied to render the character to achieve anime-like shading and highlighting. In addition, since animators tend to touch up the rendering results to reflect their intension and preference, the pipeline provides them with several ways to intuitively edit the obtained rendering results. Animator-oriented shading and highlighting is thus achieved semi-automatically with minor user-interactions.

As for shadowing in Anime [Pub_1], an instant shadow generation system for 2D animation especially Japanese style Anime is proposed. Traditionally in 2D Anime production, the entire elements including shadows are hand-drawn therefore it takes

much time to complete. To improve this problem, an easy shadowing system is developed to enable animators to easily create a layer of shadow and its animation based on the character's shapes. The system is both instant and intuitive. First, shadows are automatically rendered on a virtual plane surface by using the Shadow Map method based on input layers such as a character layer and an object layer. Then the rendered shadows can be edited by simple operations and simplified by the Gaussian Filter. Several special effects such as blurring can be applied to the rendered shadow at the same time. Compared to previous approaches, this approach is more efficient and effective to handle automatic shadowing in real-time.

They are applicable to a wide range of situations of Anime production. I believe these techniques can be applied to help animators and reduce time, labor and cost in Anime production pipeline.

Acknowledgments

First of all, I would like to express my sincere appreciation and gratitude to Prof.Morishima, my PhD advisor, and Prof.Yu and Dr.Anjyo to give me the opportunity to stay in UIUC and to learn a lot about computer graphics world. In addition, Prof.Seah and Dr.Tian for giving me the opportunity to explore and research into the interesting work to develop a professional system "CACAni System" and for their instruction, criticism and encouragement throughout the research process.

Their enthusiasm and rigor to research work greatly benefit me. The accomplishment of this thesis would not have been possible without their dedication, valuable knowledge and guidance. I am grateful for the financial support of this research provided by Waseda University.

I would like to extend my appreciation to Prof.Hashimoto, Prof.Komatsu, Prof.Kondo and Prof.Ukai, my thesis sub-examiners. Moreover, I would like to thank Dr.Yotsukura, Mr.Maejima, Dr.Iwasawa, Mr.Uemler, Mr.Kazama, Mr.Nakajima, Ms.Kasai, Ms.Nakatani, Prof.Uchiyama, Ms.Sato, Ms.Tanaka, Prof.Kohmura, Prof.Kurumizawa, Prof.Mao, Prof.Nakajima, Prof.Tai, Mr.Sawada, Dr.Baxter, Ms.Kimura, Mr.Nishimura, Mr.Shimotori, Mr.Ishikawa, Mr.Shiraishi, Mr.Sekine, Mr.Sano, Mr.Sugimori, Mr.Ito, Mr.Ogata, Mr.Takahashi, Mr.Ohmuro, Mr.Ohashi, Dr.Qiu, Dr.Hahn, Mr.Kyota, Mr.Jia, Mr.Lu, Mr.Liew, Mr.Low, and Ms.Desiree who have helped me to develop the technique, system, and approach, and to create demo animations and images, in addition broadened my knowledge in the field through the period of research.

Last but not least, I would like to thank my family members and friends for their support, encouragement and blessings.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background, Motivation and Objectives

Video contents of Japanese Cartoon Animation so-called Anime has been taking the world by storm. The TV stations in The United States are broadcasting Anime everyday and the characters' in Anime are quite popular in East and South East Asia. Thus, the demand for Anime is becoming increased year by year. However, it does not necessarily imply that the Anime production companies can handle the increased demand and it causes for the poor working conditions for animators, for the increasing the cost, and for other problems. To devise the innovative idea, therefore, Anime production companies and research institutes have recently establish a collaborative relationship. In terms of giving a feedback from animators to the researchers and then solving it, the business-academia collaboration can be considered as an epoch-making experiment. It is expected to create a breakthrough in the future.

From magnificent Disney productions to popular Japanese Anime, 2D animations, or cel animations, bring joy to both children and adults. As mentioned before, however, making traditional 2D animation is time-consuming and labor-intensive, which mainly consists of the following steps: story boarding, character design, drawing (keyframes/in-betweens), inking/painting and compositing [1]. To produce a smooth motion sequence, senior animators first draw keyframes depicting the extreme postures of motion. Assistant animators then draw in-betweens according to the keyframes [2]. Among various procedures, inbetween drawing is most tedious. Although the expanding of digitization in the animation production pipeline progressed, the digitization is just partially replacing the traditional ones and the process by hand-drawing is still considered as the typical style of Anime creation pipeline and has developed the most efficient way to achieve the quality. It takes up a large proportion, approximately 60%, of the total labor in inking/painting [3]. If in-between creation can be automatically generated by using computer, or even partly, a huge amount of time and labor can be reduced. It will significantly eliminate the production cost and collaterally improve animation quality. On the other hand, it is suitable to let computer handle the work since inbetween drawing requires less artistry. This is basically from the production way that animators draw both keyframes and in-between frames based on their taste, preference, and intentions. There are a lot of physical contradictions between frames but the

animation sufficiently attracts audience. Being able to reflect their artistic taste is considered as the advantage of the Anime production and it is the most difficult part that can be achieved by computer graphics techniques.

In terms of the Anime-like hair motion, the approach can apply to both photorealistic expression and Anime-like (Non-photorealistic) expression but is rather suitable for the latter expression. Figure 1-1 shows an example of inconsistencies in the number of strands, or locks of hair at the same character [Pub_13] This is the reason why we are motivated to research Anime hair motion creation. Hair plays an important role in the effect of real anime productions. Therefore we heavily focus on animating anime character's hair motion in this paper. There are several reasons why hair animations are so important. The first reason is that hair serves as meaningful artistic and story functions in anime. For example, hair animation could show that natural phenomena (e.g. wind is blowing at a character, Figure 1-2 (a)) are happening. Another example, Figure 1-2 (b) demonstrates that hair animation could also be utilized to express character's emotional excitement. Thus, animators could make character's hair stand on end to express surprise, anger and so on. Finally, hairstyles are utilized to define a character, and to distinguish different characters from one another. For example, even though characters may have the same basic face, their hairs give audience the impression that they are actually different characters (Figure 1-2 (c)).

In spite of its importance, animating hair has been specialized work. Although attractive hair motion is often seen on TV and in movies, there are few animators who can achieve such a sophisticated hair motion because only skilled animator can create such an engaging animation [Pub_13]. At the same time, even though they are skilled, drawing such an impressive hair animation takes a lot of time. Because of these backgrounds, giving animators a method to create hair animation between key frames can be a substantial contribution. Thus, hair motion creation in anime is a challenging task in computer graphics. To focus on this challenging task, we therefore develop a method that can produce an automatic in-between creation based on key frames by using our simulation model. Our final goal is to enable animators to easily create desired motion for characters with decent quality in a shorter time than by hand-drawing. In effect this meant that we needed a method that would produce controllable and automatic hair motion, while allowing the animator to express their creative vision. Our method requires only several hand-drawn key frames of characters as inputs.

"Anime" has its unique recognizable style and a wide range of audience not only in Japan but also around the world. As a special art form, anime can have a wide variety of the visual styles according to an animator, a production studio, and even a production.

Although such a variety exists, there are several commonalities which can be viewed as the characteristics of anime. Shading style is one of the features distinguishing anime from other cartoons.

As illustrated in Figure 1-3(a), the styles of highlight and shade are the essential elements of Anime-like shading, which are utilized to emphasize the lighting, depth, material and even the emotion and feelings of the character. As shown in Figure 1-3(b), animators usually draw several kinds of lines to create highlight, shade, etc. In Anime, highlight and shade are rendered not only to depict the relationship between characters or object and light sources. They portray the geometry information such as flat or non-smooth, and the material information such as shiny or matte. In addition highlight and shade can be utilized to grab audience's attention to characters and scenes. In fact, highlight and shade play an important role [53], [68] to represent where the light source is and at the same time they can express the character's emotion and feelings, and in 2D animation. Audience is able to recognize how the character feels from the shading and highlight which is not related to the actual lighting information (Figure 1-3 (c)).

Despite the importance and usefulness of highlight and shade, [53] animators might roughly draw them due to limited human resources and time constraints in production. Since to draw detailed highlight and shade in each individual frame requires a substantial amount of time [23], animators frequently do not have enough time to draw it with their artistic taste.

The objective of this research is ultimately to reduce animators' workload and to improve labor effectiveness for the production pipeline. These are achieved to extract more information from keyframes and incorporate it into inbetween generation for hair motion and to utilize the existing information for rendering shadow automatically. In both cases, the input is the material that has already existed in the production pipeline so that animators can utilize the techniques without changing their working procedures. Figure 1-4, Figure 1-5, Table 1-1, and Table 1-2 show how the techniques are applied to the pipeline and support their working procedures respectively.

Figure 1-1, Example of the inconstancy of hair looks (Left character),
and another example of the inconstancy of hair numbers (Right Character)

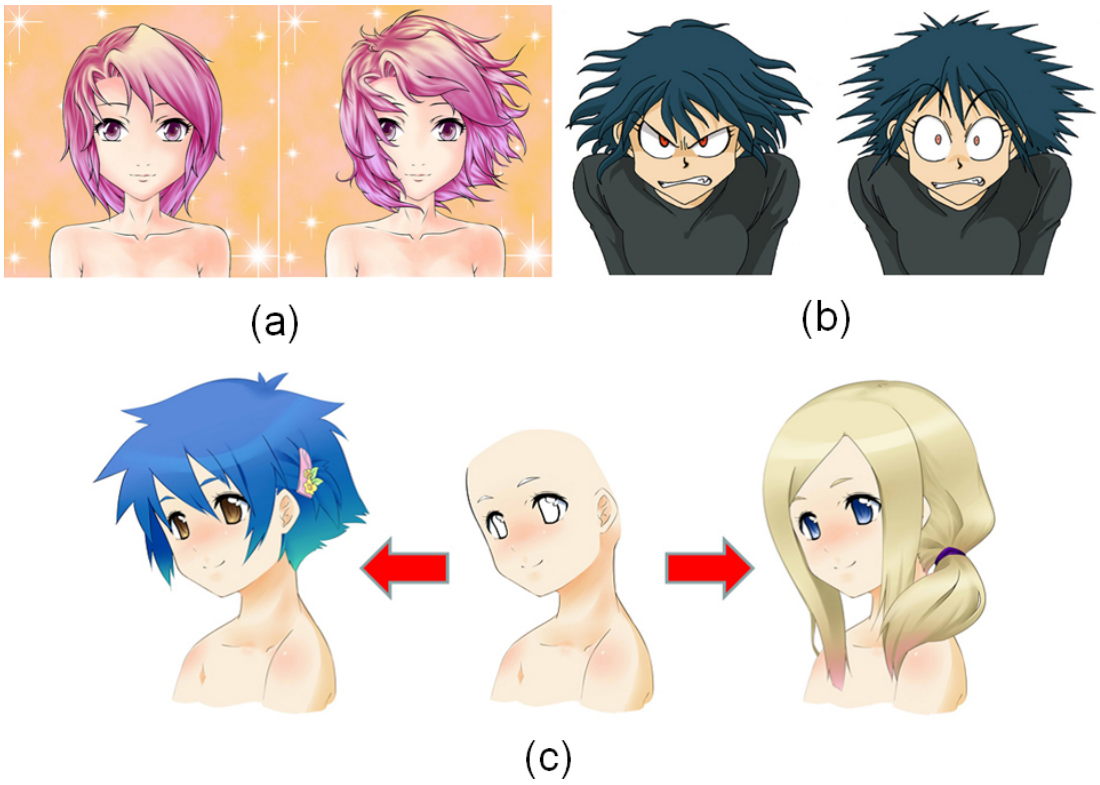

Figure 1-2, Role of Hair Expression in Anime:
(a) The character is blown by wind.
(b) Example of character's emotional excitement. Left: Anger, Right: Surprise.
(c) These characters have same faces with different hairstyles.

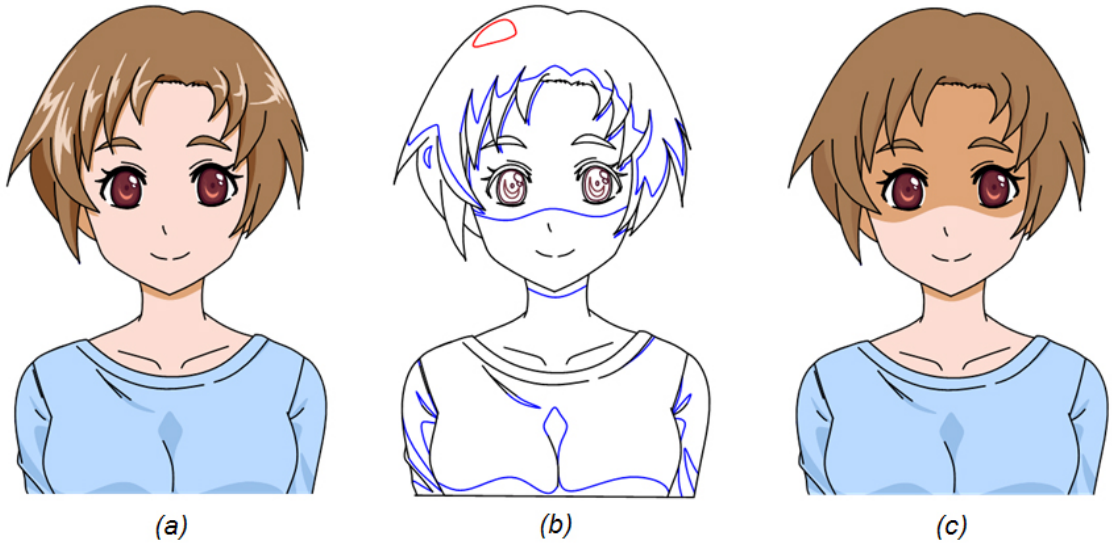

Figure 1-3, Role of Shading in Anime
(a) Anime-Like Shading Result,
(b) Example for support lines for highlight and shade.
(c) Shading for representing negative emotion

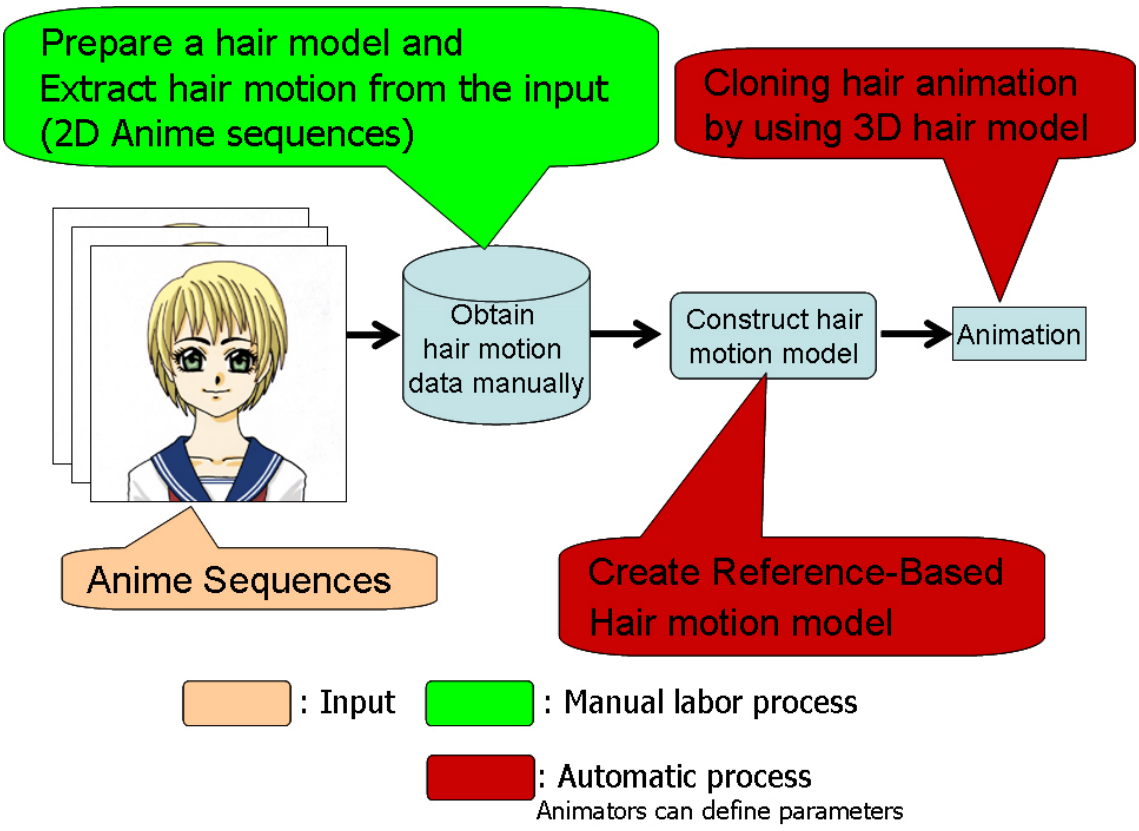

Figure 1-4, How this Approach Works in the Pipeline in Hair Motion Creation

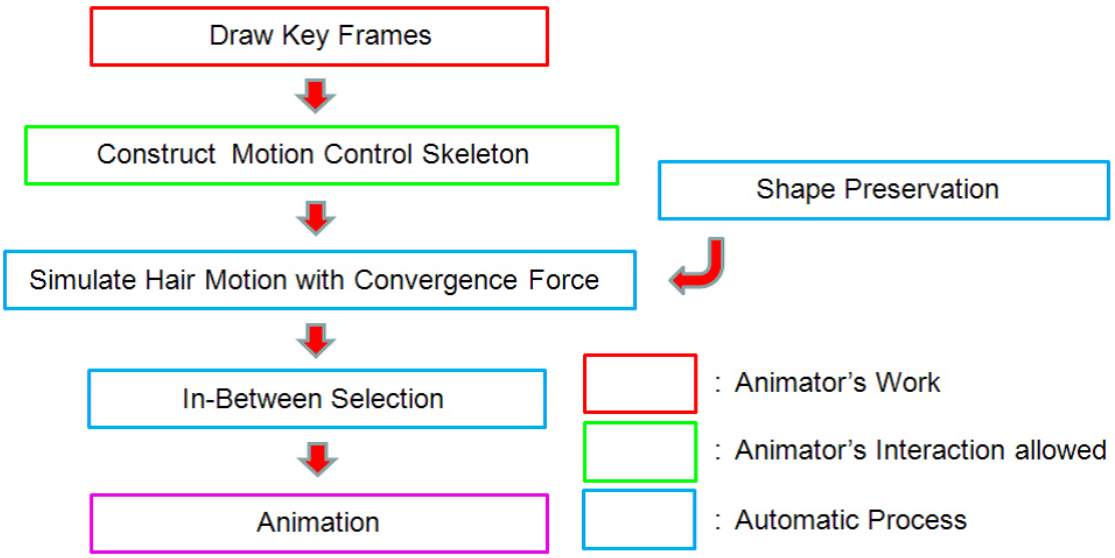

Table 1-1, Process Overview for hair in-between creation:

First, animators draw several key frames. Second, the skeleton structure for motion control is constructed. Then simulation with shape preservation is applied to hair strands. Finally In-between frames with the feature of the motion are selected according to the number of frames which animators specify.

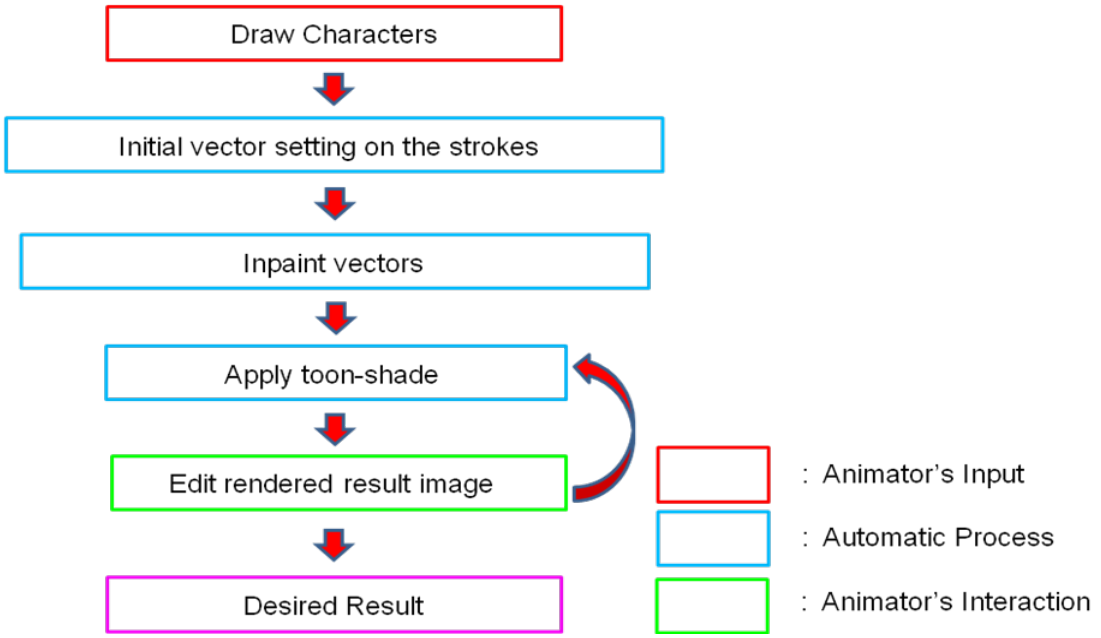

Table 1-2, Overview of the shading pipeline

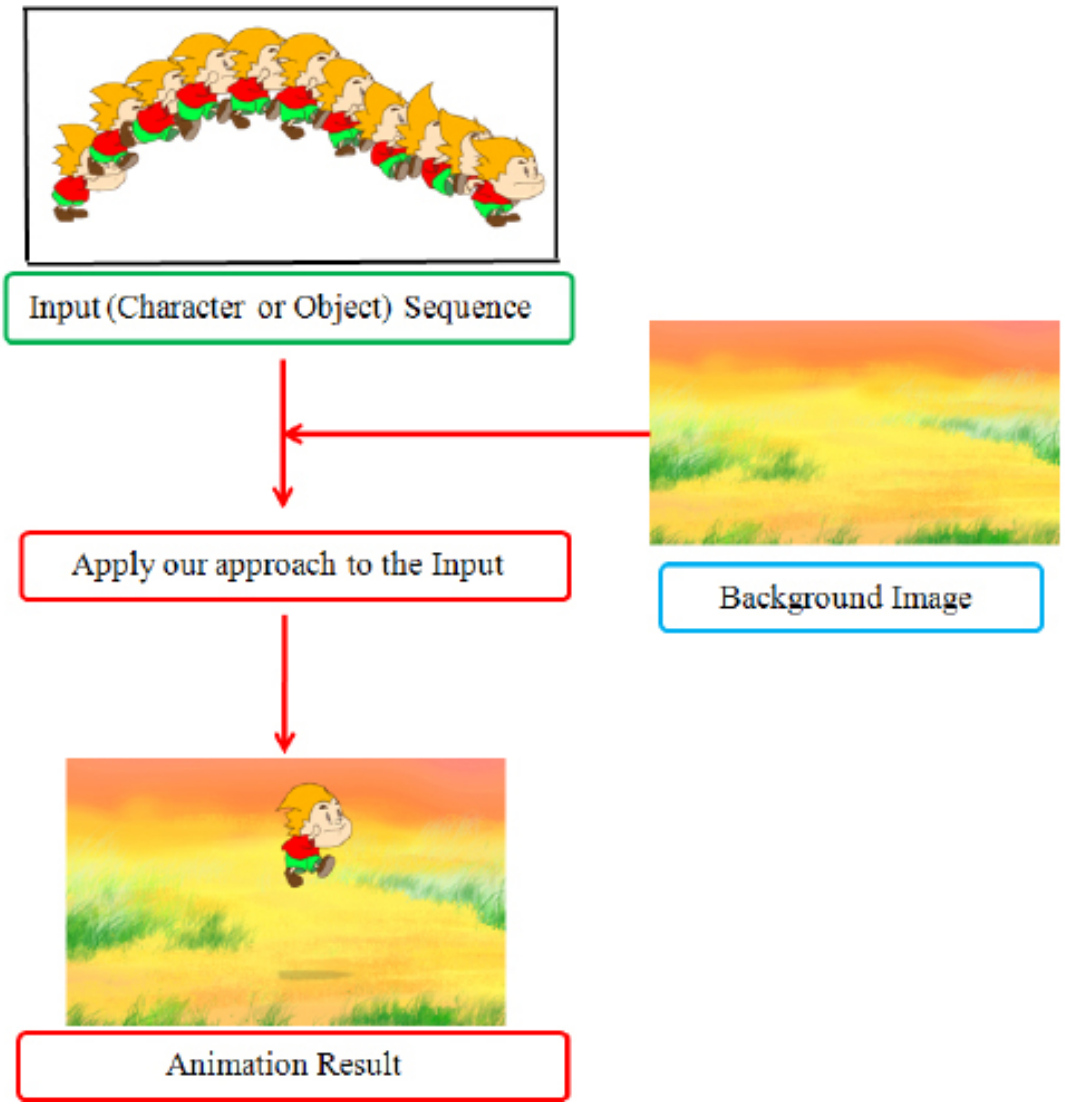

Figure 1-5, How this Approach Works in the Pipeline in Shadowing Animation

1.2 Approach Overview

One way to reduce an animator's workload is to consider hair structure in three dimensions [Pub_13]. Obviously, the advantage of using 3D model is easy to be able to re-draw, re-create, and re-model. Although Anime production is achieved in 2D structure, we could consider an Anime character as three-dimensional structure for creating hair motion easily. On the other hand, it is very difficult to express inconsistency of 2D expression peculiar to "Anime-like" expression using a 3D model.

As for reference-based hair motion creation, the approach is to create Anime hair animation, one that allows the animators to create attractive Anime hair animation using existing character animation sequence [Pub_13]. We demonstrate how to generate cartoon hair animation that is accentuated in "Anime-like" motions. The advantage of this method is that animators are able to choose the existing the character's hair animation to apply environmental information such as winds to another character that has a 3D structure. In fact, animators can re-use existing Anime sequences as inputs to give environmental information to another character as if the character exists in the same scene. 3D character's hair motions are created based on hair motions from input animation sequences. First, users extract hair shapes at each frame from input sequences, and then, construct a physical-based hair motion model from extracted hair shapes. The Anime-like hair motion is created using the motion model.

This paper also describes simulation-based in-between creation for hair animation. First, animators need to draw key frames of characters. Based on these inputs, this technique needs to create the skeleton structure in each key frame. Next, how to apply the simulation model to each hair strand is described. Since the simulation model has a convergence force to the target hair strand, the motion converges on the target keyframe automatically. The details of the motion model and convergence force are described in 3.2.31, 3.2.32 and 3.2.33 respectively. After the system has obtained in-between frames, the system automatically reconstructs the motion according to the number of frames that animators specify with the feature of the motion. Thus, the simulation-based approach is achieved.

In addition, automatic Anime-like shading for animator's drawing is handled by boundary normal analysis. In fact, based on the hand-drawing lines, the virtual normal vectors are estimated by TV-Stokes model. And then, using the estimated normal vectors, the 2D hand-drawn image is rendered as animators want such as Anime-Like, bumpy surface, and reflect the surrounded environment.

In terms of interactive and intuitive shadowing, this approach needs only layers of a character or an object animation sequence as inputs. First, animators draw layers of a character or an object animation sequence. Then, because these inputs are transparent except for the character or object region, shadows for the characters or object are rendered automatically by using the Shadow Map. Generally, the Shadow Map is utilized for a 3D object. In our approach, the Shadow Map is applied to the characters' 2D shape in the input animation sequence, with transparency. Several shadow shapes can be designed by editing the rendered shadows. The editable parameters are the level and gradient of the ground where the shadows are cast, the position of the light source, and the simplification and blurring parameters for abstracting shadow shape. The approach enables animators to produce Anime-like shadow animation by editing these parameters.

1.3 The definition for “Anime-Like” [23], [49], [50], [68]

In this paper, the term “Anime-Like” is utilized as a keyword several times. Therefore, “what Anime like” has to be defined in this sub-section. First, the definition of the term “Anime” is explained as Japanese Animation. Cartoon Animation is in the broad sense of the term of Anime. In fact, since Japanese style of Cartoon Animation has a significantly-distinguishable feature from other Cartoon Animations, Anime has become one of the genres of Cartoon Animation. Therefore, the definition of the term “Anime-like” here can be replaced with the expressions that are observed characteristically in Anime. In fact, we define the term that characterizes the styles of the Anime. The following lists are the examples of Anime-Like expressions that can be seen in several Anime contents.

In terms of Character’s Motion, Transitions, and Actions

- Follow-Through, Anticipation
- Squash and Stretch
- Abrupt change (Physically discontinuous), but looks natural
- Keep or converge the original Shape in the end or a key-frame

In terms of Rendering

- Fill in a single color in a region
- Show the emotional factor with shading and highlighting
- Tell audience geometrical information more than physical correctness

Although there are a lot of styles of Anime-Like expressions, all of the expressions cannot be handled by one universal way. Therefore, the paper especially describes the Anime-Like hair expression in Chapter 3, and the Anime-Like Shading and Highlighting, and Shadowing in Chapter 4. In Chapter 3, the expression is considered as the character’s motions, Transitions, and Actions. In this paper, the featured expressions are “Abrupt changes” and “Keep the shape”. In Chapter 4, the expression is considered as the Anime-Like Rendering. The featured expressions in this Chapter are “Fill in a color in a region” and “Tell the audience geometrical info”.

1.4 Thesis Organization

This chapter 1 explains the background, the motivation, the objectives and the overview of our research.

Chapter 2 briefly reviews 2D animation production process, commercial 2D animation systems, and existing animation techniques by other researchers for inbetween generation, hair animation, and shadowing techniques. In the following chapters, our research work on computer-assisted hair motion creation and shadowing technique for 2D animation is described in detail.

In Chapter 3, we explain our proposed approaches for Anime-Like hair motion creation. There are two ways to create the hair motion. One is physics-based hair motion creation, the other is reference-based hair motion creation.

In Chapter 4, automatic shading and highlight for hand-drawing image and interactive and intuitive shadowing techniques based on the character animation sequence are introduced, The UI system is also described in this Section.

The results are shown and discussed in Chapter 5. Then what the approach has achieved is described.

Conclusions and future work are made in the last chapter.

Chapter 2: Surveys and Related Work

2.1 Animation Production Process [90], [91]

It is necessary and important to have an overview of the workflow in traditional animation production and to understand how traditional animators conventionally works their tasks before developing approaches that will really meet the needs. In the following parts of this section, the conventional steps of 2D animation production are explained, among which keyframe and inbetween drawing process is described in detail.

2.1.1 Traditional Animation

Traditional animation is also referred to as cel animation, or hand-drawn animation. In a traditional animation, each frame is drawn by hand. The major steps to make a traditional animation [2], [4] are illustrated in Table 2.1 and will be described as follows. For higher efficiency, there is often an overlap in the above procedures and even parallelism in practice [2].

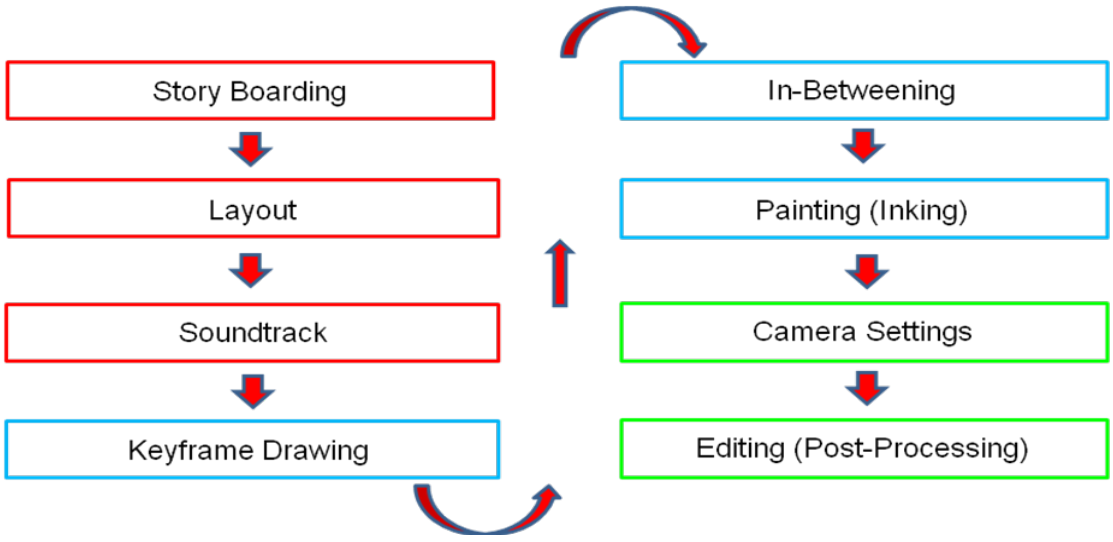

Table 2-1, Overview of traditional animation production process

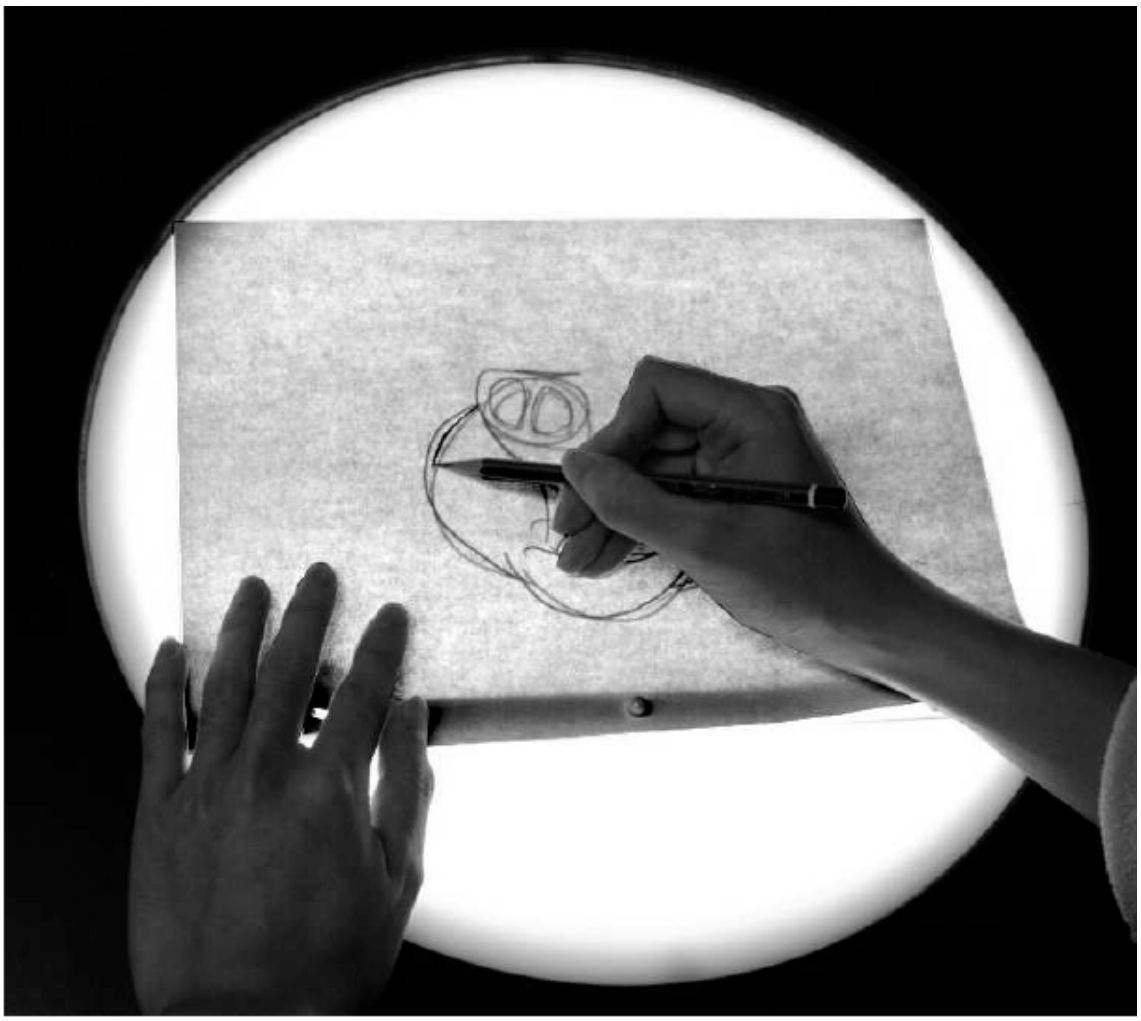

Figure 2-1, Drawing frames Over Light Box

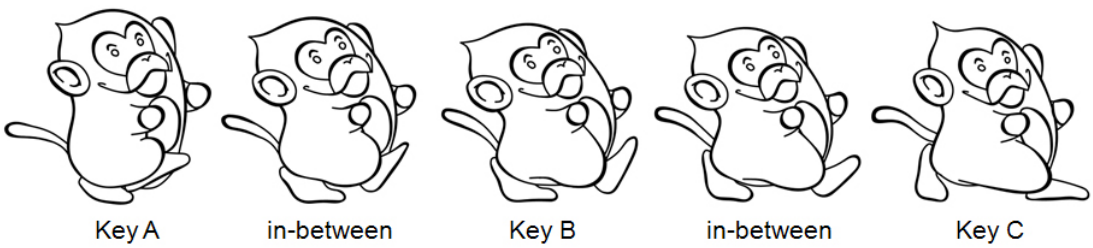

Figure 2-2, Keyframes and In-Betweens

Story Boarding

To depict the story which the animation is to tell, synopsis and scenario play the role of a summary and a detailed text respectively. The storyboard offers a visual description of the film's key moments, composed of a sequence of drawings that define specific actions.

Layout

Based on the storyboard, this step mainly consists of character designing and action plotting. It is important to illustrate the appearance of individual character in different poses and from various perspectives as well as the proportion between each character. This will serve as a reference when the entire task of drawing is later distributed to a team of animators so that the drawings from different animators remain consistent.

Soundtrack

In conventional animation, soundtrack is usually recorded preceding the animating process. Motion must be analyzed frame by frame in order to achieve synchronization with dialog and music. Alternatively, scoring and sound effects can be added in final editing and mixing.

Keyframe Drawing

The drawing of frames is the spirit of animation. In this process, animator draws sequences of animation on sheets of paper perforated to fit the peg bars on the light box. A light box, also called the trace box, as shown in Figure 2.2, holds a fluorescent light fixture that shines up through a rotatable disk of frosted glass or milky-white plastic. This allows animators to see through the sheets and easily trace images on a sheet of paper below the one he is working on. The peg bar on the disk helps to register the sheets with corresponding holes.

A keyframe animator will draw keyframes in a scene, using the character layouts as a guide. They draw enough frames to get across major points of the action. A frame often comprises multiple layers. Different characters and backgrounds are usually drawn in different layers so that characters may move without disturbing backgrounds, or backgrounds may vary by scrolling to one side as characters move. Only what moves in each frame must be drawn anew. All the layers will be composited into a single frame. During this stage as well as the subsequent inbetween drawing process, line testing will be carried out, where the pencil drawings are quickly photographed or scanned and synced with necessary soundtracks, in order to get a preview of the actions and rectify

flaws. An animator may be required to re-work on a scene many times before the director approves it.

In-Betweening

Once the keyframes are drawn, the drawings go through a clean-up process and are forwarded to the in-betweeners. The in-betweeners will fill up remaining frames which are placed between keyframes in order to make the movement look smooth. These frames are called in-betweens. In-betweeners create in-betweens according to the changes between keyframes. To draw an inbetween frame, the given two keyframes and a blank sheet for in-between are first registered by holes onto the pegs of the light box for alignment. The pegs maintain registration so that all the keyframes and the inbetween are in exact relationship to each other. The light shines from the back of the sheets making the sheets look transparent so that the drawing on all the sheets can be visualized. The in-betweener examines the change between the keyframes and draws the mid-elements of corresponding points, lines and shapes based on which he draws in-betweens. Once finished, starting from the first keyframe, every frame in the sequence changes slightly from the previous one until finally the second keyframe is reached. An example is illustrated in Figure 2.2. The resulting drawings are again pencil-tested for approval.

Among various procedures of traditional animation production, inbetween drawing is one of the most tedious ones. A 90-minute cel animation consists of more than 100,000 frames, a large proportion of which are in-betweens. Automation of inbetween generation would allow artists to leave this tedious work to the computer and concentrate on the more creative work of drawing keyframes. However, quality is a crucial issue. As recapitulated by John Halas [5], one of the world's most famous animators, “movement is the essence of animation”. Animators must make sure that objects move in a convincing manner. In-betweens play an important part in the quality of the final animation. Despite the computerization of some other procedures in the production, so far in-betweens are mostly drawn manually. The results of some automatic inbetween generation systems do not reach the standard of the industry.

Painting (Inking)

In early years, the completed pencil drawings are subsequently transferred to acetate cels by xerography. The inking and painting stage is gone through on the cels with the right degree of opacity. Nowadays it is a common practice to scan the drawings into computer and paint them using a wide variety of software tools.

Camera Settings

After compositing all the layers of a frame and carefully checking the action in the scenes assuring that everything is properly executed and identified, all the frames are brought together on the rostrum camera, illuminated and photographed onto color films or videotapes frame by frame at certain rate, e.g. generally 24 frames per second in Anime production.

Editing (Post-Processing)

To transform the shot film into a final product, post-processing and editing is applied where the film is assembled, sorted and spliced to polish the final production. Contemporary in anime production, this procedure is executed by other software, e.g. Adobe AfterEffect.

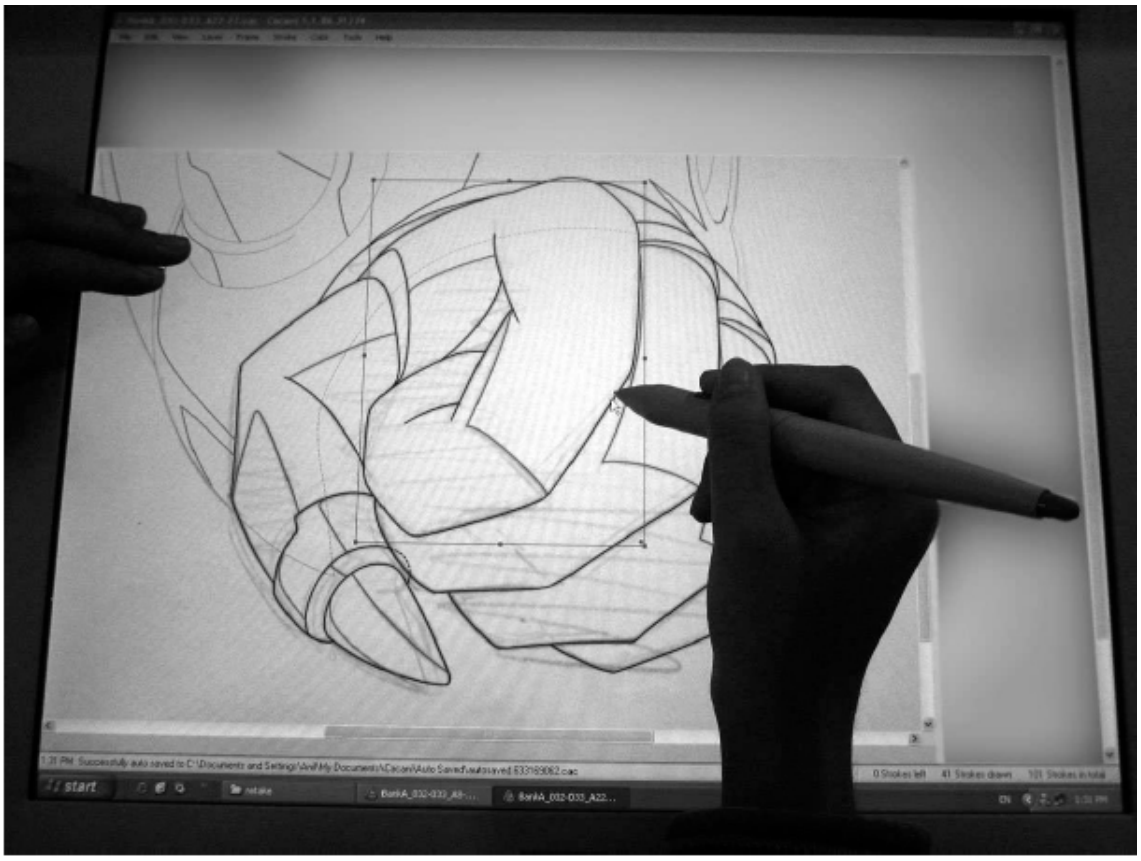

Figure 2-3, Drawing frames on a Tablet PC

2.1.2 Computer-Assisted Animation

With the development of computer techniques, the process of traditional 2D animation production gradually moves to the computer-assisted area. Recently, quite a few procedures are being executed by using computer animation systems. For example, inking and painting are usually implemented digitally after the outline drawings are scanned into the computer instead of being transferred to cels and then colored by hand. Post-Processing such as compositing and editing is mainly implemented on computer as well. Using computers facilitates easier exchange of artwork and cooperation of production between departments, studios and even countries.

For the keyframe drawing process, many animators still utilize pencil and paper. The drawings are scanned into computer and stored as raster images for further processing. On the other hand, it has also become possible for animators to directly draw frames in computer using a digital tablet, as shown in Figure 2.3. It is similar to the way of drawing on papers but easier for modification. The drawings can be stored as either raster images or vector form.

Raster images capture the exact styles and characteristics an animator desires and it is easier and more intuitive for a traditionally-trained animator to control the appearance of the final work. Animators can produce higher-quality drawings using their paper drawing skills, free from possible limitations imposed by vector drawing tools. The process is more straight-forward. However, the higher quality is demanded that requires larger resolution of image, the more storage space is needed for the image files.

Vector drawings are easier and faster for modification. They have advantages, e.g. scalability and compactness. They can be scaled and rotated without losing fidelity, which allows easy re-use by simply changing output resolution. Thus vector form is concise in representation is of small storage size. It is more suitable than raster drawings for Anime production to be displayed on high definition display devices, which are becoming more popular and might be dominant in market.

Several techniques are also developed to convert raster drawings to vector form which further facilitates the drawing process. It is likely that paperless 2D animation will supersede the traditional pencil-and-paper animation, not too far in the future.

No matter whether an animator uses raster or vector drawing, it is difficult to generate in-betweens automatically. In most of cases, in-betweens are still drawn frame by frame manually. Therefore it is one of our major goals to make automatic inbetween generation easier and more accurate.

2.2 Commercial 2D Animation Systems [90], [91]

There are many commercial 2D animation systems available to assist animators in the actual production. In this subsection, several systems are referred as concentrating on the modules or techniques for frame drawing and in-between creation.

As described in [6], the two main classes of automated in-betweening systems are template-based [7, 8] or explicit correspondence [9]. A designer creates a template for each character in template-based systems and then animators need to restrict the character's movements according to the template. The correspondence between each keyframe and the template must be specified. Explicit correspondence systems utilize two keyframes and generate in-betweens on a curve-by-curve basis. An operator is also required to specify the correspondence between the curves in the two keyframes. A zero-length curve is created occasionally when a new stroke appears or disappears between keyframes.

[Tic Tac Toon]

Tic Tac Toon is an application for professional 2D animation productions produced by Toon Boom Technologies Inc. It is the first animation application which utilizes vector-based painting and sketching that replaces the traditional paper-based production process. It provides interface enabling professionals to draw and sketch as same as they do on paper, and transforms a pen trajectory with varying pressure into a stroke of varying thickness using Bezier curves and least squares curve fitting, in real-time. However, it does not provide a computer-assisted inbetweening module. Animators need to draw in-betweens frame by frame according to keyframes as they do in traditional way. Tic Tac Toon has been discontinued by Toon Boom Technologies Inc.

[Toon Boom Studio] [10]

Toon Boom Technologies Inc. later developed a sequence of animation software packages, including Toon Boom Studio for individuals and Toon Boom Harmony, an enterprise version for studios, advanced from their previous product “US Animation”. With integrated workflow and configurable user interface, their systems cover most of the tasks in animation production and are good for producing paper-cut animations. The products are vector-based, resolution-independent, supporting vectorization from scanned bitmaps or directly drawing on tablet. Harmony is armed with more powerful features, such as morphing, inverse kinematics and its glue effect. In-betweens can be created using the morphing tools. However, it is object-based, i.e. user creates a

vector-based object and transforms it to create the next keyframe. The practice is limited to relatively simple movements. It is quite a different solution from traditional animation where animators draw keyframes separately. Here animators only create the first frame and alter it by some transformation to be the second keyframes. Then the system interpolates or morphs the transformation to generate in-betweens. It is not intuitive and sometimes troublesome and complicated to control and it can only generate limited motion.

[Animo] [11]

Animo is software for professional animation production which integrates 2D and 3D animation. It is produced by Cambridge Animation Systems and was firstly launched in 1990. Animo utilizes a mixture of bitmap and vector technology to provide both clean, fast, able to animate vector paint and the accuracy and essence of the original drawings though bitmaps. Animo utilizes the original scanned lines from animators' drawing to perform the subsequent processes such as painting, which is raster-based. Vector painting and drawing tools allow users to create inbetween frames but only camera and element positions can be implemented keyframes and automatically created in-betweens.

[Retas! Pro] [12]

RETAS! (Revolutionary Engineering Total Animation System) Pro is a digital animation production package produced by CELSYS, Inc. It is the leader software utilized in Japan. RETAS! is primarily a bitmap based system. One of its components, TraceMan, is the scanning application that allows users to scan their images into the system and trace the images so that the outlines are suitable for painting. It previously utilized conventional raster representation for traced strokes. In the recent version, it also supports vector tracing. After loading a raster image to be traced, user clicks on the lines to drop control points and the system automatically interpolates the control points into vector representation. However, no inbetweening module is available.

[Micromedia Flash] [13]

Micromedia Flash is well known vector animation software for web animation creation. It enables users to create vector-represented objects in keyframes. The parts of a character with different movement are defined as independent objects and manipulated separately. User creates keyframes by dragging the object to its starting and ending location in the keyframes. The moving path can also be specified. Flash will draw and

transform the inbetween shapes.

Because of the limitations of the software and small le sizes for web distribution, a large part of animations created in Flash are simplistic limited animations, or in a cutout animation style. Several typical features of Flash animation could be: jerky natural movements (walk, gesture), lip motion without interpolation, abrupt changes from front to profile view or even no head turns at all. Generally Flash is utilized for quick and less complicated animation production which does not require high quality as traditional animation has shown.

[Mirage Studio] [14]

Mirage Studio is a paperless digital animation system designed by Bauhaus Software, Inc. It is a bitmap-based system, which digitally produces frames similar to those drawn on real paper. It utilizes bitmap to produce stylish brush effects, change opacity, and accurately simulate natural-media tools, such as crayon, charcoal, oil painting, chalk etc. It can apply motion along linear or spline paths similar to the in-betweening function in Flash.

2.3 Related Techniques

2.3.1 Hair Animation [Pub_4, Pub_13]

This overview of related work is limited to previous work on hair dynamics, focusing on explicit hair models. These models consider the shape and dynamics of each strand. While they are especially suitable for the dynamics of long hair, they do not consider cartoon simulations. Anjyo et al. [15] utilized a simplified cantilever beam to model hair and utilized one-dimensional projective differential equations of angular momentum to animate strands. Rosenblum et al. [16] and Daldegan et al. utilized sparse characteristic hair to reduce computation time. Kim and Neumann [17] presented an artful method for creating hairstyles that utilizes Multi-resolution Hair Modeling (MHM) system, which is based on the observed tendency of adjacent hair strands to form clusters at multiple scales. Yu et al. [18] also presented a method for creating hairstyles. These advances greatly improved hair expression in computer graphics.

Several researchers have proposed novel approaches to hair-to-hair interaction. Hadap and Magnenat-Thalmann [19] proposed modeling dense dynamic hair as a continuum by using a fluid model for lateral hair movement. Hair-to-hair collision is approximated by the pressure term in fluid mechanics while friction is approximated by viscosity. Hair-to-air interaction is approximated by integrating hairs with an additional fluid system for the air. Chang et al. [20] modeled a single strand as a multi-body open chain expressed in generalized coordinates. Dynamic collision for hair-to-hair is solved with the help of auxiliary triangle strips among nearby strands. The input to their simulation algorithm is an initial sparse hair model with a few hundred strands generated from their previous hair modeling method. Plante et al. [21] proposed a "wisps model" for simulating interactions in long hair. Bando et al. [22] proposed a method in which they model unordered particles that have only loose connections to nearby control points. By freeing particles from some constraints, they are able to animate hair including hair-hair interactions at a reasonable computational cost. Considering hair-hair interaction is also making significant contribution to hair expression in computer graphics.

In terms of cartoon expression, Lasseter [23] is very likely the first paper to describe the basic principles of traditional two-dimensional hand-drawn animation and their application to three-dimensional computer animation. In this paper, he clearly describes cartoon animation and what it requires of an animator. Paul Noble and Wen Tang [24] achieved cartoon hair modeling and animation by using NURBS Surfaces to model the primary shape and motion of cartoon character hair.

Rademacher proposed the method that a three-dimensional structure is utilized in cel animation.[25] The reference hand-drawn image of the object or character often contains various view-dependent distortions that cannot be described with conventional 3D models. Therefore, to prepare view-dependent models, which consist of a base model, a set of key deformations created by the base model, a set of corresponding key viewpoints, and a given discretional viewpoint, they interpolate the key deformations that are specific to the new viewpoint. They thus capture the view-dependent inconsistencies of the reference drawing.

In terms of using the data captured from 2D images, there are several research have been achieved. Y.Wei [26] et al have presented to model hair geometry from images taken at multiple viewpoints. They have achieved impressive result inputting about 30 to 40 views. K.Zhou [27] et al have proposed the method to apply non-rigid and exaggerated deformations of 2D cartoon characters to 3D meshes.

2.3.2 Shading Techniques & Surface Reconstruction

Recently, 3D computer graphics techniques are utilized in the Japanese "Anime" industry (e.g. Ghost in the Shell and Innocence). The movie Appleseed is a landmark anime movie featuring hyper realistic imagery and a hybrid 2D and 3D style [69].

A lot of research has been proposed to derive 3D shape from 2D primitives, which can roughly be classified into two categories: geometry-based and image-based methods. Methods in the former category reconstruct a 3D surface or object from 2D primitives by inflating a closed curve [70] or inferring 3D free-form surfaces from visible contours [71]; Methods in the latter category only interpolate surface normals for an image based on given sparse ones [51].Our method in this paper is inspired by the Lumo system [51], and it belongs to the latter category as no 3D rotation for the character is needed in the whole shading pipeline.

One of main ideas in our paper is to propagate the vectors into the domain from initial vectors on the boundary of domain and several curves inside of the domain. This idea was originally introduced in image inpainting [72], [73], [74], [75], [76], [77], [78]. As a real inpainting artist usually makes a smooth connection of orientation of isophote from outside into inside of inpainting regions, it is crucial to obtain smooth extension of directional information. Unlike to many literatures in image inpainting, the two-step algorithm in [74], [75] divides into estimating tangent vectors for isophote and reconstructing images from the tangent vectors. In this paper, we prefer to call the former step as vector inpainting. As the images are considered as surfaces, the incompressibility conditions for estimating vectors in the inpainting domain become very crucial and it allows to inpaint large regions [75]. Since we have large regions for inpainting vectors, the incompressibility conditions are necessary to achieve better inpainting results.

To obtain dense vector fields from spare vector fields was recently introduced [51], [79], [80], [81] and the authors have successfully applied the fields to surface reconstruction and single view modeling. In [79], a smart method for assigning the initial vectors along curves and editing the dense vectors are proposed based on a minimization of a functional with L2-norm which allows obtaining smooth dense vector field. The main issue in the dense vector field [79], [80] is that it does not satisfy the compatibility conditions. In our paper, we propose an energy minimization with the compatibility constraints to obtain both more natural surface reconstruction and anime-like shading.

Toon Shading has been popular way of expressing the Cartoon-like rendering. It has

shown in a variety of 3D rendering software, video games, and animations. The idea is quite simple, at the same time powerful. The shading utilizes a texture that represents how the final shading is handled from the basic color to the darker color than the basic.

Barla et al [82] proposed a novel idea for the extended toon shader. They described two extensions to the basic algorithm for handling view-dependant effects they utilize a 2D texture which can handle so-called "toon-detail" than can vary according to the depth and surface orientation. The other extension was to extend the toon shader for the use of a modified normal field which can range from the initial normals to simpler set of normals taken from so-called abstracted shape. They basically developed a concept "level of abstraction" in perspective of toon shading. Although their concept is sophisticated, we adopt the basic toon shading to achieve an anime-like shading result.

Anjyo et al. [53] developed a system that can render stylized highlights on 3D object utilized in cel animation. Several user-friendly operations were implemented in their system to modify the highlight vectors. As an extension method to [54], users can directly manipulate stylized light and shade in real-time with explicit light source control. A set of simple algorithms enables the interactive controlling of highlight and shade by intuitive operation such as click-and-drag. Therefore, users have a more straightforward way to control highlight and shade than tuning the parameters. Our editing approach follows their concept for reference. Shimotori et al. [52] proposed an interactive way to directly edit shade by adjusting the surface normals on the triangles by simple mouse operations. The idea of changing the surface normals facilitates the rendering of other frames as the normals are not affected by the camera movement. In terms of editing concept, I also referred to their ways. However, this editing system is more intuitive, providing animations several editing methods.

2.3.3 Shadowing Techniques

Shadow mapping has been developed by Williams [28] as an image-based shadowing technique. Since it utilizes existing hardware functionality such as texturing and depth buffering, the technique is particularly suitable for hardware implementation. Hardware accelerated shadow mapping has been implemented by Segal et al [29]. This method is fundamental for Shadow Map. Their method does not consider the view point so that there is an aliasing problem in close-range shadow rendering from the view point. The following methods try to solve this problem.

PSM (Perspective Shadow Maps) [30] solved the problem for the close-range shadow rendering. While the resolution for the close-range shadow from the view point has been solved in PSM, there is a problem that the resolution of the long range shadow rendering from the view point. LSPSM (Light Space Perspective Shadow Maps) [31] improved on PSM by providing a solution for the problem. TSM (Trapezoidal Shadow Maps) [32] is more advanced. Handling the projected light space as trapezoid, they exploit the shadow map area effectively. The fundamental of this method is to create convex hull by using 2D Quickhull. They define the trapezoid that is perpendicular to the line by connecting near surface and far surface and that has the upper base surface and lower base surface of meeting of the top of convex hull. Furthermore, SSM (Subdivided Shadow Maps) [33] is an improved method for TSM. TSM's transformation is applied to each surface of cone in light space. LogPSM [34] (Logarithmic Perspective Shadow Map) is a parameterization for reducing perspective aliasing. They derive the aliasing error equations for both point and directional light sources in general position. Then they parameterize a simple combination of a perspective projection with a logarithmic transformation. LogPSM can reduce aliasing error significantly and save both storage and bandwidth respectably.

As for the implementation of soft shadow, PCF (Percentage Close Filter) is well-known. VSM (Variance Shadow Maps) [35] is an improved version of PCF. Normally only the depth value is stored in Shadow Map. However, VSM stores the depth value and the square value of the depth. Additionally in VSM, the Gaussian filter can be applied to shadow texture in same way as applied to color texture. Therefore, this method can handle the filtering process for each shadow map so that it is much more efficient than PCF.

Several user interfaces have been proposed for editing shadows in computer graphics. Nakajima et al. [36] developed the system called "Kagezou" for easy shadowing in Anime production pipeline. We refer to their shadowing approach for the input 2D

image plane and inherit their user interface for editing. Though the overall concepts are similar, our system is specialized for CACAni System's production pipeline. Petrovic et al. [37] proposed an innovative interface that creates Anime-like shadows semi-automatically. Our system follows their key concept of shadow generation. While their method requires an amount of time to create shadow animation, our system enables CACAni users to create shadows in the entire sequence instantly. Pellacine et al. [38] developed a user interface that enables animators to edit shadows directly on the editing screen. Like their interactive interface, we implement our user interface that provides CACAni users with intuitive and simple mouse operations. Decoro et al. [39] created a user interface that can be utilized to design several forms of shadows for rendering non-photorealistic shadows. We are influenced by this concept and develop our system that simplifies shadows as they appear in Anime. Nakajima et al. [40] developed a tool that takes advantage of both 3D and 2D techniques. Their goal was to edit shadows for Anime-like expression to 3DCG model intentionally. Conversely, our method focuses on rendering instantly and editing intuitively shadow throughout the 2D input sequence quickly using 3DCG techniques.

Chapter 3: Hair Motion Creation

3.1 Reference-Based Hair Motion Creation

3.1.1 Overview

The novelty of this method is that animators are able to choose the existing character's hair animation to apply environmental information such as winds to another character that has a 3D structure. In fact, animators can re-use existing cartoon sequences as inputs to give environmental information to another character as if the character had existed in the same scene from the beginning. The 3D character's hair motions are created based on hair motions from input animation sequences. First, animators extract hair shapes at each frame from input sequences, and then, build up a virtual equation from the differences between extracted hair shapes each frame. The Anime-like hair motion is created by using this equation.

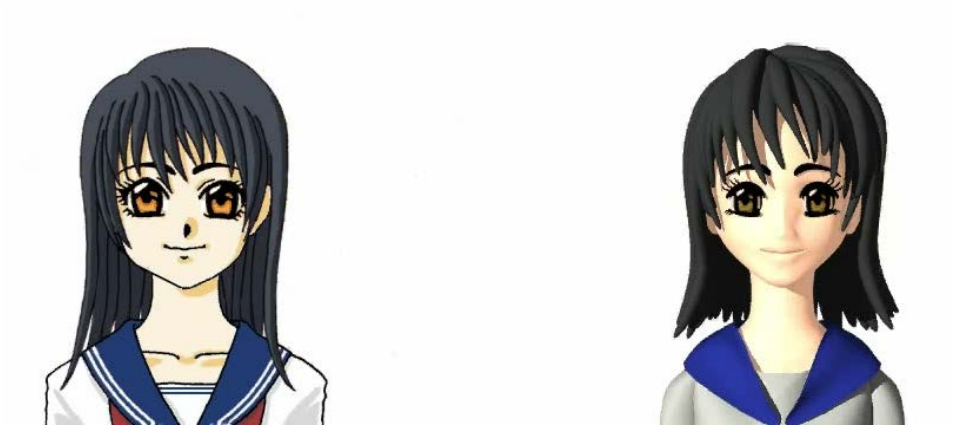

Figure 3-1, Character's Model
(Left: Input hand-drawn, Right: Created 3D character's model), the 3D character model is constructed by a 3D modeler manually based on the original hand-drawing character.

3.1.2 Prepare Data for Hair Motion

As mentioned above, the hair motion in Anime is sometimes contradictory to a physical law. This is because an animator draws animation sequences based on their touch rather than a physical law. In this paper, we extract hair motions from existing cartoon animation sequence to apply animators' touch to a 3D hair model. Therefore, before starting to obtain hair animation data, we need to model a 3D character similar to the character drawn by hand-drawing. (Figure 3-1)

3.1.21 Hair Structure

It is difficult to calculate hair motion accurately. In our method, therefore, hair motions are modeled approximately. When the external force to the hair is not so large, it can be assumed that hair length hardly changes. Also the expression in Anime, we postulate the characters' hair length hardly changes in a case like breath of winds. Thus, each hair is expressed as concatenation of rigid body in this approach (Figure 3-2). This is common expression for hair in computer graphics [15]. Based on this model, we re-create "anime-like" hair motions with a 3D character's hair model.

Anime hairs tend to be expressed as a thick object compared with real human hairs. Our hair model has boundary lines to form the hair shapes and a centerline to control hair motion (Figure 3-4). Also, in order to treat animation with camera motion, we need to construct the hair model as an object with thickness like a circular cone.

3.1.22 Hair Motion Model and Basic Equations

Figure 3-2 Shows that our model of hair strands are constructed by n-th control points and (n-1) segments. The following equation shows the i-th motion equation denoted by spherical-coordinate system. Here r represents the length of the segment, (i-1) is the local origin point.

$$\begin{aligned} x_i &= x_{i-1} + r\sin\theta_i\cos\varphi_i \\ y_i &= y_{i-1} + r\sin\theta_i\sin\varphi_i \\ z_i &= z_{i-1} + r\cos\theta_i \end{aligned} \tag{3.1}$$

the 0th control point is considered as the root, it assumes that the position of the point never moves. Figure 3-3 shows the example of spherical-coordinate system. In this

paper, we assume that each segment is independently moved in ⊿t considering the previous control point as a fulcrum point. Since we utilize the spherical-coordinate system, the movement is a rotation based on the fulcrum point.

Based on the assumption, I_0 is the inertia moment around the point, N is the moment applied to a segment. Here ω is an angular velocity. Following equation presents the motion equation for our 3D model

$$I_0 \frac{d\omega}{dt} = N \tag{3.2}$$

The elements in the spherical-coordinate system are the length ***r***, the angles θ、φ. Here the length ***r*** never changes in our system, the equation (3.2) denotes following equations.

$$\begin{aligned} I_\varphi \frac{d^2\varphi}{dt^2} &= N_\varphi \\ I_\theta \frac{d^2\theta}{dt^2} &= N_\theta \end{aligned} \tag{3.3}$$

Here, I_φ 、I_θ are the inertia moment of each element. In addition, to solve this equation of motion in computer simulation, we utilize the Runge-Kutta method. The equation 3.4 shows the basic form of the Runge-Kutta method. The method reduces errors to obtain the average value with correcting the variation in ⊿t.

$$\begin{aligned} &\phi(dt) = \dot{\phi}(dt)dt \\ &\dot{\phi}(dt) = \phi(0) + \frac{1}{6}(k_1 + k_2 + k_3 + k_4) \\ &\begin{cases} k_1 = N_\phi(0)dt \\ k_2 = N_\phi(0 + k_1/2)dt \\ k_3 = N_\phi(0 + k_2/2)dt \\ k_4 = N_\phi(0 + k_3)dt \end{cases} \end{aligned} \tag{3.4}$$

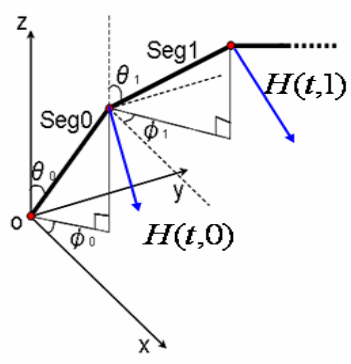

Figure 3-2, Rigid Body Model

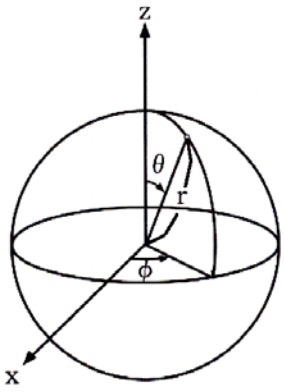

Figure 3-3, Spherical-Coordinate System

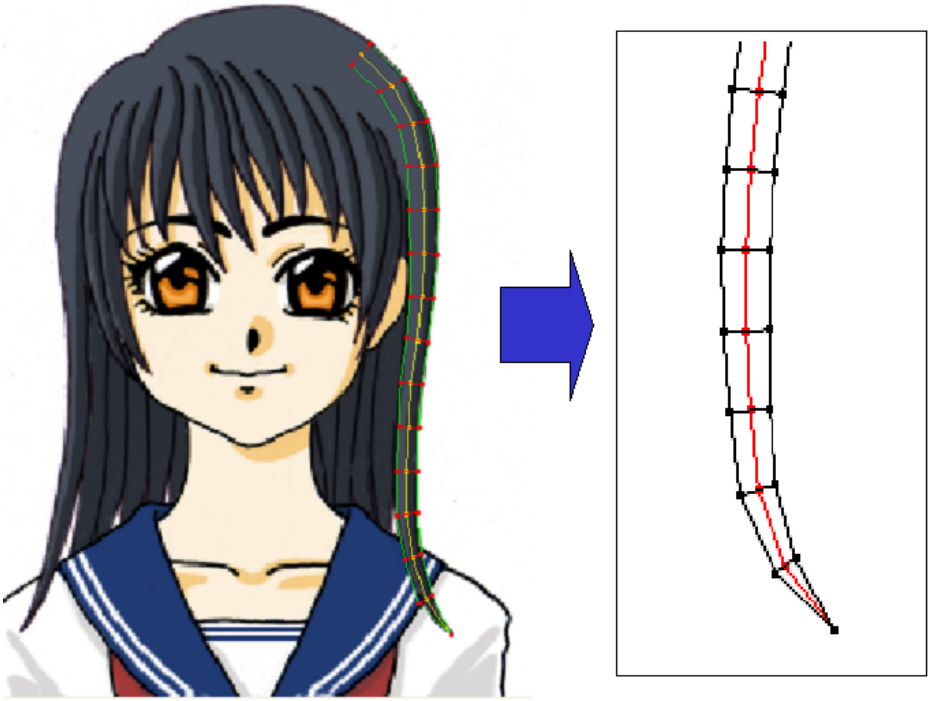

Figure 3-4, How an Animator Extracts the Hair Shape

3.1.23 Obtain Motion Data

Based on the hair model prepared in section 3.2.21, hair motion data is captured by the difference between each frame. Hair shape of the character from animation sequence drawn by animator fits in the prepared 3D hair model. The procedure of this process is determination of the hair root position first. Then, adjusting the connection angle, each segment length, and each strand thickness. This procedure is repeated until the hair shape is fit to drawn character. (Table 3-1)

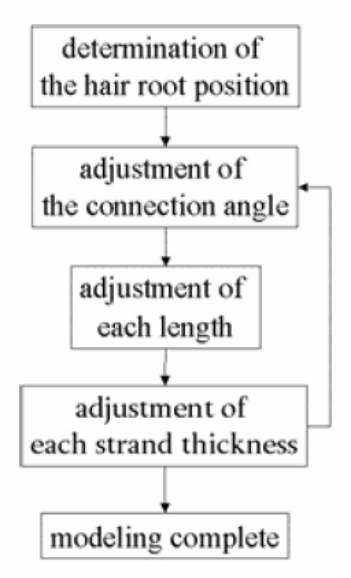

Table 3-1, Modeling Steps

3.1.3 Modeling for Hair Motion

3.1.31 Estimate the External Force

From the data captured in section 3.1.23, the external force to the hair is estimated. Why the external force in the scene needs to be estimated is described in Section 5.1.2 "Application". We estimate the external force to the hair from the difference of the angle between each frame. The external force can be obtained from the difference at every frame for each node of each hair strand in the existing animation. Obtaining the external force makes users possible to construct reference-based motion model. Therefore, we can re-use obtained external force to another character. Since the model can be considered as a physically-based motion model, we can utilize these equations such various situations. The motion model is expressed by following equations.

$$\omega_T = \theta_T - \theta_{T-1}$$

This is a general expression for an angular velocity ω in our simulation model. Where, ω_T is an angular velocity at frame T, and θ_T is an angle at frame T.

$$\ddot{\theta}_{ij} = \frac{(\theta_{ij}(t+s) - \theta_{ij}(t)) - \omega_{ij}(t-s)}{s} \tag{3.5}$$

Where, $\ddot{\theta}_{ij}$ is the angular acceleration, i is the number of hair strand, j is the joint number from the hair-root, and ω is the angular velocity. Here, we suppose that angular acceleration is constant between frames.

$$I_{ij}\ddot{\theta}_{ij} = H_{ij} \tag{3.6}$$

$$I_{ij}\frac{d^2\theta_{ij}}{dt^2} = H_{ij} \tag{3.7}$$

Equation 3.6 is the equation for each joint. Where I_{ij} is the inertia moment, $\ddot{\theta}_{ij}$ is

angular acceleration, and H_{ij} is the torque (rotation moment). Equation 3-6 can be denoted as equation 3-7. Since the spherical coordinates are utilized to model the hair, the motion of the depth direction (z-direction) can also be controlled by using the virtual equations. So, the method can handle animation with camera motions. This is the huge advantage of using these virtual equations because the input animations sequences have obviously two-dimensional data.

3.1.32 Define the hardness of hair motion

We have obtained the external force in the scene in Section 3.2.31. This makes users to be able to construct a reference-based motion model at each joint for a hair strand. However, having physical equations at each joint of all hair strands would be too much information. It could be utilized the equations formed by all joints. It of course makes result animation more precise reproduction to the input cartoon animation. Since one of the purposes of this research is to reduce users' workload, however, extracting all hair strands is against our policy. So, the method requires users to determine representative points as control points to reduce input information. The external forces at each joint become equalized between control points. (Figure 3-5) Although the external force is discontinuity from the hair root to the end from the beginning, the external force becomes discontinuity more radically at a control point. This makes hair motions unnatural. Therefore, the external forces at the joints that are not chosen as a control point are interpolated by spline functions using control points. (Table 3-2, Table 3-3)

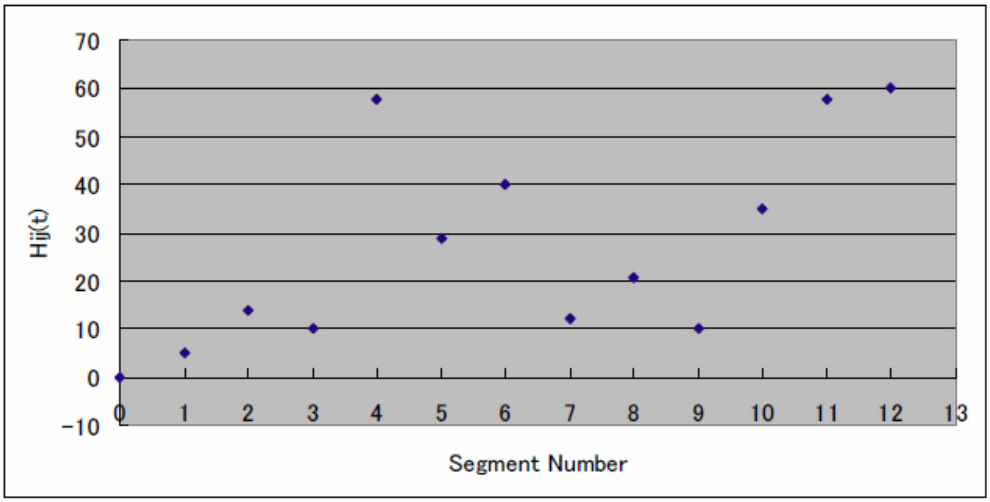

Table 3-2, Original External Force Data Distribution

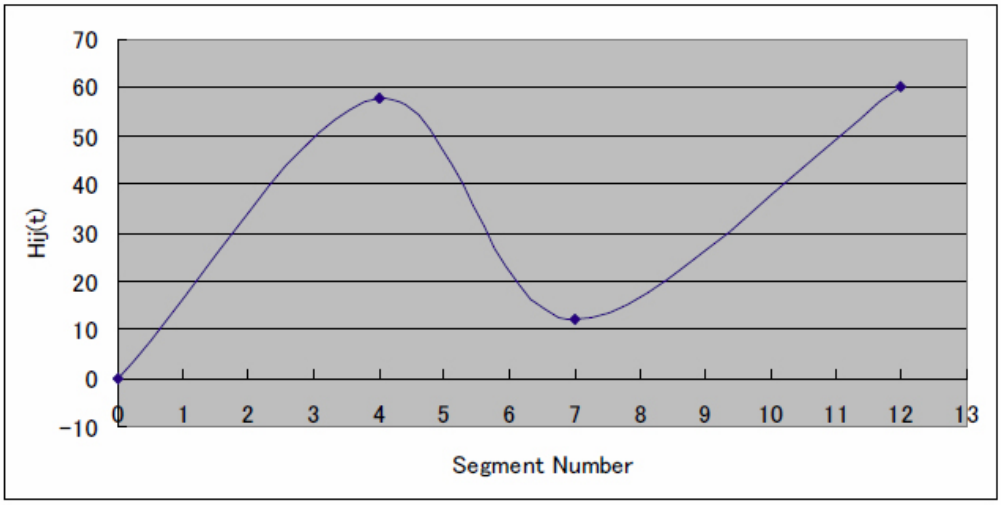

Table 3-3, Interpolated External Force Distribution

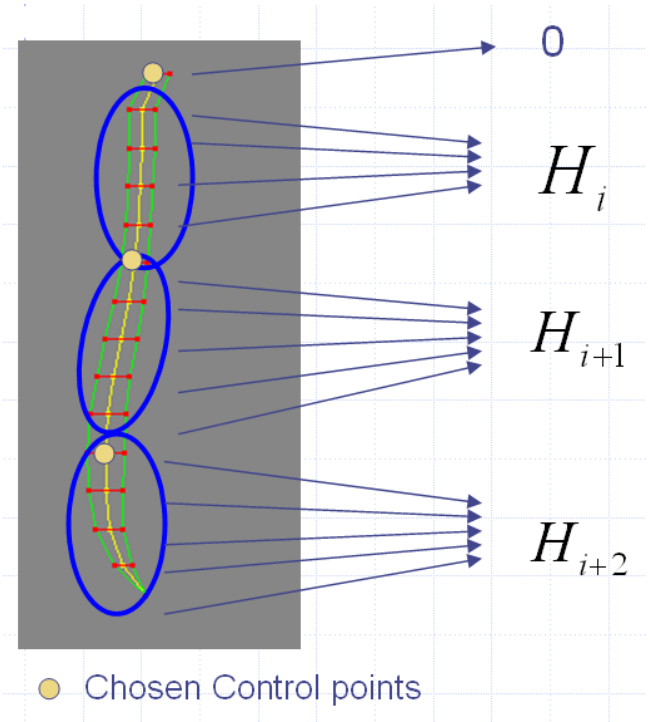

Figure 3-5, Example of how the Torque is Divided

In addition, how animators determine the representative points should be the factor of the stiffness of hair motion. Animators can controllably define which joint of original data is utilized to construct the simulation model. In fact, how animators determine which control points are utilized would be a factor of hair stiffness. If an animator choose the control points densely around the hair root and sparsely around the end (Figure 3-6(a)), hair motion becomes harder than the original motion. On the other hand, hair motions become smoother if an animator chooses more points near hair ends. (Figure 3-6(b))

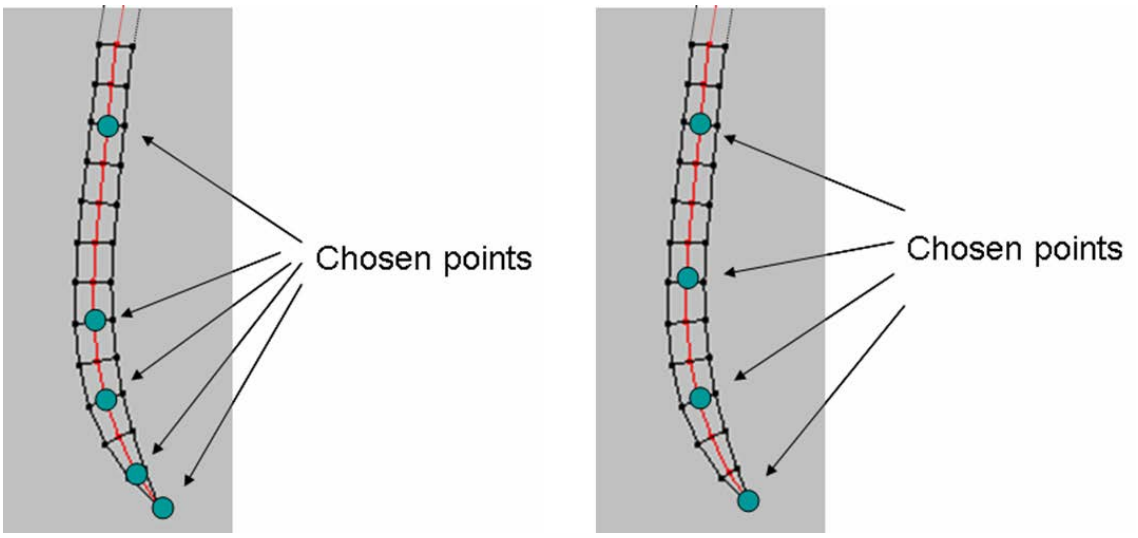

Figure 3-6, How to choose the Control points
(a) choosing points densely around the end (left)
(b) choosing points sparsely around the end (right),
The hair root and the end must be chosen

3.1.33 Parametric Control for the hair stiffness

Since the external force has been estimated from the animation sequence, the virtual equations can be applied to another character as if that character were in the same environment from the beginning in the animation. However, there is no guarantee that another character has same hair style such as hair length, hair shapes and the number of hair strands. Even though the character is in the same scene, therefore, the hair should move differently. We consider that the hair stiffness should depend on the length of hair strands. Consequently, the hair stiffness that depends on how long the hairs are should be controlled by a user parametrically. Equation 3-8 represents how to define the hair stiffness.

$$I_{ij}\frac{d^2\theta_{ij}}{dt^2}=\frac{1}{L}H_{ij} \tag{3-8}$$

Where, L is the factor of the stiffness. L becomes larger; the hair looks harder. L becomes smaller; the hair looks smoother and softer. Generally, if the hair becomes shorter, it moves harder. And if the hair becomes longer, it moves softer and smoother. User can direct depending on their sense.

3.1.34 Define the Restoring Force

In the real world, it hardly can be seen that the hair style is retained after the wind blows to the hairs. In the Anime world, on the other hand, the characters' hair style is always kept after applying the any effect, even though strong wind blows to the characters' hair. Because keeping the character's appearance is the one of the Anime-Like features. Especially for hair style, it describes the characteristic of the character so that the hair style should be retained after motion.

In order to keep the hair style, the restoring force is defined to keep characters' hair style. Our approach utilizes a rigid body model to express the hair motion and supposes the length of the hair is not changed. Therefore, the initial angle is stored first and then the difference of the angle between each time step is defined as an elastic force. This force is such as a spring force. Thus, the angle becomes far from the original angle, the force becomes larger and this makes motion change abruptly.

$$I_{ij}\frac{d^2\theta_{ij}}{dt^2}=\frac{1}{L}H_{ij}-k(\theta_{i0\,j0}-\theta_{ij}) \tag{3-9}$$

Where, k is an elastic coefficient and θ is the initial angle of the joint. Using equation 3-9, motion suitable for the scene is achieved.

3.1.35 Animation

Hair rendering is not the focus of this paper. However, we need rendering to produce visually appealing results. Since our technique is for cartoon characters' animation, we implement cartoon shading [46].

Since Anime characters' hair tends to have a thick object, we need to give a thickness to the hair strands. Hair strands shape is interpolated by NURBS surface. (Figure 3-7)

Also, collision detection is important factor to show the animation attractive and convinced. We implement hair to head and hair to body collision detection. However, dynamic hair-to-hair collision is not achieved.

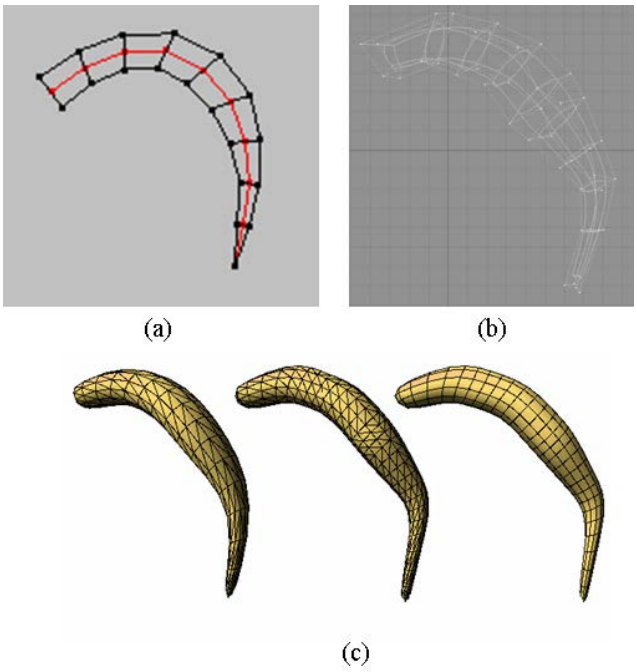

Figure 3-7, How to Create Hair Thickness
(a) Shaped from the input animation sequence, (b) Creating hair thickness by NUBS surface, (c) Applied result

3.2 Simulation-Based Hair Motion Creation

3.2.1 Overview [Pub_13]

A method for creating Anime hair animation that easily retains the "Anime-like" aspect has been developed. The ultimate goal is to produce an interesting hair animation that matches the hair designs shown in hand-drawn key-frames utilized in the actual production pipeline. Our method is a hybrid one taking advantages from both interpolation and physical simulation. The input is a sparse set of hand-drawn key frames for an Anime character in the keyframes.

The technique reduces time needed for the production by creating automatic in-betweening based on proposed simulation based model. As mentioned, the inputs required are several key frames for the character's hair drawn by animators. A crucial step in this approach is building a simulation-based motion model.

First, the skeleton structure on the hair strands is automatically constructed by a contour-based extraction method. Next, for the several situations configured by animators such as windblown hair or swinging hair with the motion of the character's head, motions of the skeleton model are created based on the forward dynamics algorithm. The technique can create the hair motion to converge onto the target hair strands by applying a convergence force to the hair motion. And then, the shape of the hair strands is deformed by the results of the motion synthesis of its skeleton. To avoid unnatural shapes such as a self-intersection during the deformation, we apply a shape maintain method to the hair strands. In-between frames themselves are automatically generated by using our simulation model in this way. Because the simulation continues until the hair motion converges onto the target hair strands, the obtained in-between frames are more or less accordingly to the animator's design. We provide a feature frame selection algorithm for animators after obtaining the in-between frames. According to the animator's specification, a certain number of in-between frames are selected automatically. Comparing to previous models, our simulation model can faithfully adhere to the input key frames, even though we utilize a simulation-based motion model. Animator-oriented hair animation is thus achieved by applying our method to each key frame.

3.2.2 Constructions of Hair Data from Original Input[Pub_13]

The method requires the preparation of various types of hair data before starting the simulation. The approach first needs to prepare hand-drawn key-frame images of the animated character (like those shown in Figure 3-8). These images are utilized as the original input. This is the only step in which a skilled animator draws images in this method.

We suppose the input hand-drawn image graphics are vector-based. Thus, we can utilize hair shape information to create the object of hair structure. To simulate hair motion, we adopt the skeleton structure. Figure 3-9 shows how to obtain the shape of the skeleton for the target hair strand. Our method requires animators to set the root points of a hair strand. Then our method can set the points onto the boundary lines based on the hair length. At this time, the number of the points between the left boundary and the right boundary must be same. After the basic setting is finished, the center line (the skeleton structure) is set in the center of the both the left and right point. In this way, animators can set the skeleton manually, but automated is desired. We describe how to create the skeleton of the hair strand in the next sub-section.

Figure 3-8, Key-Frame Images
(Copyright by Anime International Co. (AIC), Inc.)

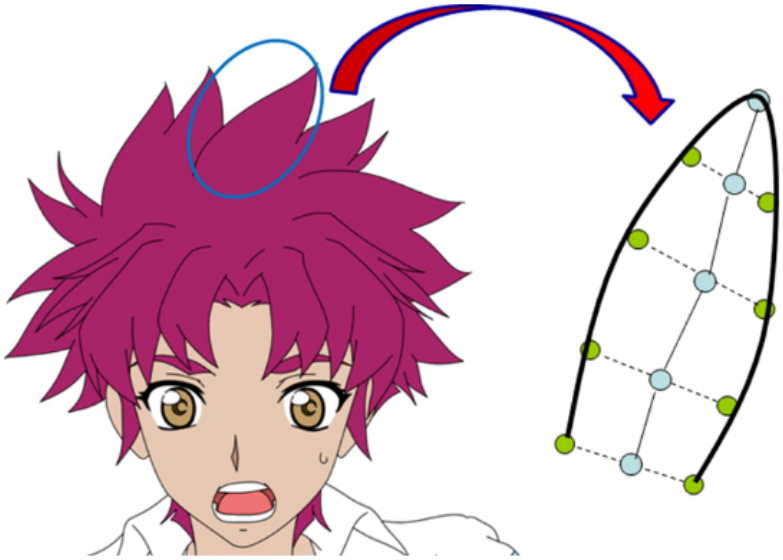

Figure 3-9, Character's Hair Strand Structure
(Copyright by Anime International Co. (AIC), Inc.)

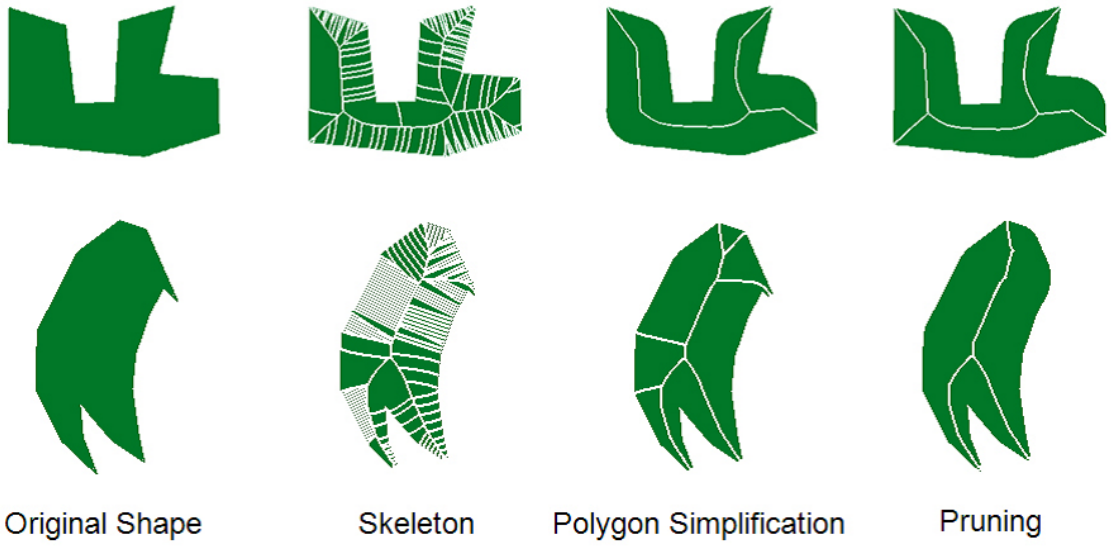

Figure 3-10, Results of Skeleton Extraction: (from left to right)
original shapes, the extracted skeletons (white lines)
without any pruning, the extracted skeletons after DCE with reconstructed shapes,
and the final pruned skeletons.

3.2.21 Skeleton Structure Construction

A chain structure for the simulation model is utilized for our simulation model. For this reason, we need a linkage that represents a trunk of hair strand to create hair motion. Therefore, the skeleton extraction based on the outlines of the hair strands drawn by animators is required. Animators need to draw at least one key frame to create in-between frames.

In our method, hand-drawn strokes which depict the character's shape including hair strands are converted into vector-form represented by a piecewise cubic Bezier curve. Therefore, the points on the strokes can be applied to create skeleton structures. Although our method can automatically create the skeleton structure at each hair strand, we allow animators to specify points on the stroke manually to let them reflect their preference. Either way, the skeleton is created based on the animator's hand-drawing. In our method, we construct a skeleton structure at each key frame from the contour lines for hair strands.

First, the contours of hair strands are rasterized to a binary image, and then an automatic skeleton extraction technique [61] is applied to the binary image. The technique consists of the following three steps.

The first step is polygon simplification. The rasterized binary image is considered as a polygon whose vertices are boundary pixels. The polygon is simplified by removing vertices iteratively. This process called Discrete Curve Evolution (DCE) [62] is achieved according to a relevance measure K given by

$$\mathrm{K}(\mathrm{s1},\mathrm{s2}) = \frac{\beta(\mathrm{s1},\mathrm{s2})\, l(\mathrm{s1}) l(\mathrm{s2})}{l(\mathrm{s1}) + l(\mathrm{s2})} \qquad (3 - 10)$$

where, line segments s1 and s2 are the polygon sides incident to a vertex v, β (s1, s2) is the turning angle at common vertex of segments s1 and s2, and l is the length function normalized in regard to the total length of a polygonal curve. The higher value of K, the larger is the contribution of the arc (s1 $\cap$ s2) to the shape. In each step of DCE, a vertex which has the smallest K is removed. The process is iterated until the shape of the polygon becomes a convex or the number of vertices is smaller than a threshold. As a result, DCE can lead to a subset of vertices that best represents the shape of a given contour and it can also be considered that the given contour is partitioned by the resulting subset of vertices.

The second step is skeleton growing. At first, the Euclidean Distance Transform (EDT) of the binary image of a given shape is computed by [63]. Then the point with the maximum value of the EDT is selected as a seed of the skeleton. Recursively, the skeleton is generated by adding points which lie on ridges of the EDT. During this Skelton growing process, for a point to be added, it must additionally have its generating point on at least two different contour segments of a given contour partition. The redundant skeleton branches are eliminated by this criterion.

The third step is a skeleton pruning. After the skeleton growing based on DCE, some redundant branches still may remain. The pruning method called Discrete Skeleton Evolution (DSE) [61] iteratively removes skeleton end branches with smallest relevance. The relevance for branches is measured as their contribution to shape reconstruction. Based on this definition of a skeleton, a skeleton point must be the center of a maximal disk/ball contained in the shape. B(s, r(s)) denotes the maximal disk centered at a skeleton point s, and r(s) denotes the radius of it. The reconstruction of a skeleton S is denoted R(S) and given by

$$\mathrm{R(S)} = \bigcup_{s \in S} \mathrm{B(s, r(s))} \tag{3 - 11}$$

The relevance of each branch path Pi is defined as follows,

$$w_i = \frac{\mathrm{A(R(S - P_i))}}{A(R(S))} \tag{3 - 12}$$

where, A is an area function. In our implementation, since the result of reconstruction is a binary image, an area function calculated by counting the number of pixels. We construct a linkage which has a tree structure from the extracted skeleton points. We adapt the DCE algorithm in order to reduce the number of joints. We assume that all joints are in the same plane and the all joint axes are perpendicular to the plane. Figure 3-10 illustrates the examples of the procedure to construct a desirable linkage structure from the contour line step by step.

Figure 3-11 represents an example of the linkage structure for our simulation model. The green points on contour lines are automatically generated based on the animator's specification. Although the points on the hair strands can be selected automatically, our method also allows animators to manually set the points for the purpose of reflecting

their preference. In addition, purple points are set based on the skeleton extraction algorithm. By connecting the purple points, the skeleton structure which is utilized to create hair motion is obtained.

Figure 3-11, Constructed Tree Linkage based on the Skeleton
(Copyright by Anime International Co. (AIC), Inc.)

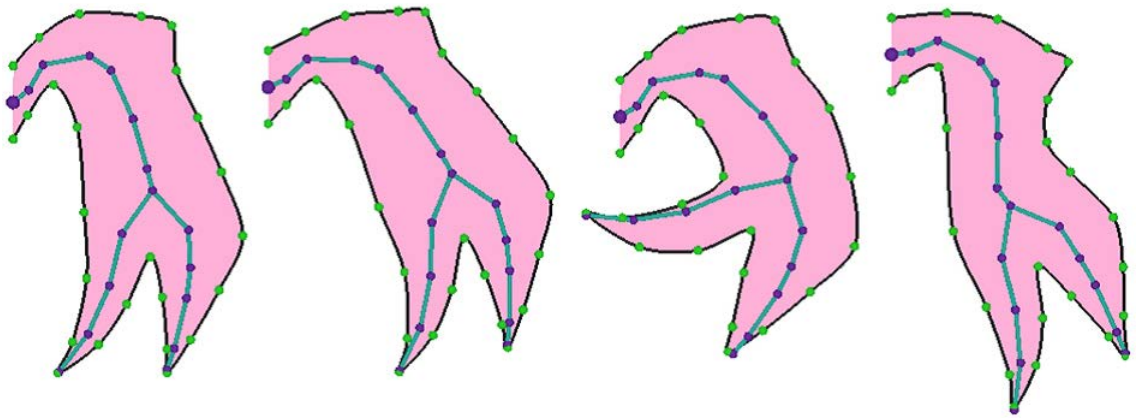

Figure 3-12, Results of Shape Deformation

3.2.22 Shape Preservation during Motion

The position of the joints on the constructed skeleton move during our simulation is applied. At the same time, the positions of the initial hair strands also move according to the motion of the skeleton. If points on the hair strands are moving by only the corresponding link, the resulting shape of the hair strands bend abruptly and self-intersections may occur. These states produce unnatural animations because real hair strands do not have a bone structure. To maintain the overall hair shape during the motion, we need to find appropriate positions of the points on the hair strands. To implement it, we apply the As-Rigid-As Possible Shape Manipulation algorithm [64] to the hair strands.

First, a triangle mesh is constructed from the points on the hair strands and the joint positions by using a Delaunay triangulation. We add a number of vertices generated in a grid pattern to the triangle mesh. The results are not strongly affected by the number of vertices because the deformation algorithm is robust against the mesh density. The vertices of the triangle mesh are classified into two kinds, the constrained vertices called “handles” and the rest free vertices. Given the coordinates of the constrained vertices, the deformation algorithm computes the coordinates of the remaining free vertices. We utilize the joint positions as the constrained vertices.

In this algorithm, pre-computation is required when handles are added or removed. The method solves the problem as a two-step closed-form minimization problem which consists of a scale-free distortion and a scale adjustment. By using quadratic error metrics, each minimization step becomes a system of linear equations and can be solved at the pre-computation phase. In our case, since the set of constrained vertices is constant, the method does not need any computation to invert matrices during interaction. Therefore it can find the new positions of free vertices very quickly.

After the positions of the points on the hair strands are acquired, the curve that represents a hair strand is rendered by using a spline interpolation. Figure 5 shows examples of the results of the shape deformation.

3.2.3 Simulation Model for Hair Strands [56]

One of the advantages of our method is that we create inbetween frames based on the simulation model. Since physical simulations are hard to control (it usually requires a parametric control so it is hard to achieve desirable postures such as shapes), it has been considered as not suitable for in-between creation. To achieve the in-between creation, we define a force named as "convergence force". Our simulation model supposes two cases; the final state of the motion is given, and the trajectory of the base of the linkage is given. When the final state is given, our method can generate in-betweens which converge on the final state. When the trajectory of the base of the linkage is given, for example, our motion model can create motion of hair strands according to the movement of the character's head.

3.2.31 Physical Properties for the Simulation Model

Our simulation model can handle gravity and wind as external forces using the forward dynamics simulation. Several parameters, such as, the spring and the damping coefficient of each joint and mass of each link, are required for our simulation model. Animators can adjust these parameters interactively via a GUI (Figure 3-13).

We consider that the shapes of all links are cylinders which have a constant radius and the mass of each link is proportional to the length of the link. The moment of inertia is defined as follows.

$$\mathbf{I} = \frac{M}{12}\begin{pmatrix} h^2 & & \\ & 0 & \\ & & h^2 \end{pmatrix} \qquad M = \pi r^2 h \tag{3 - 13}$$

where, the axis of the cylinder lies on the Y axis, and h is the length of the link and r is the radius of the cylinder. Since the moment of inertia is proportional to the cubic of the length of the link, the spring and damping coefficients should be also proportional to the cubic of the length. We divide the linkage into three parts, and animators can set different parameters for different parts. The spring and damping coefficients of the joint i are calculated by using the user-specified parameters K_s, K_d and K_{part} as follows.

$$K_s^i = K_{part} K_s h_i^3 \tag{3 - 14}$$

$$K_d^i = K_{part} K_d h_i^3 \tag{3 - 15}$$

The torque exerted on the joint *i* is calculated as follows.

$$\tau_i = -K_{part}^i (\theta_i - \theta_i^{init}) - K_d^i \omega_i \tag{3 - 16}$$

where, θ_i and $\theta_{init}\, i$ are the current and initial angle of the joint, respectively, and ω_i is the angular velocity of the joint.

When a wind is blowing, the external force exerted by the wind on a link is calculated by Eq.(3-17) [65]

$$F^{wind} = \rho W L V^2 \sin\phi \tag{3 - 17}$$

where, ρ is the density of air, W is the width of the link, L is the length of the link, V is the wind velocity, ϕ is the angle between the link and the wind direction.

We apply the soft joint constraint [66] to a joint angle to limit the range of motion of joints. When the joint angle exceeds the range of motion, the constraint torque is generated as a virtual spring and damping. The torque generated by the spring and the damping at a joint *i* is calculated as follows.

$$\tau_i^{const} = \begin{cases} -K_s^{const}(\theta_i - \theta_i^{HiLimit}) - K_d^{const}\omega_i & if \quad \theta_i > \theta_i^{HiLimit} \\ -K_s^{const}(\theta_i - \theta_i^{LowLimit}) - K_d^{const}\omega_i & if \quad \theta_i < \theta_i^{LowLimit} \\ 0 & otherwise \end{cases} \tag{3 - 18}$$

where, θ_i is the current angle of the joint, and the range of motion is from $\theta_i^{LowLimit}$ to $\theta_i^{HiLimit}$, and K_s^{const}, K_d^{const} are the coefficient of the virtual constraint spring and damping force, respectively.

When the trajectory of the base of the linkage is given by user, the spatial velocities and spatial acceleration of the base are calculated and the forward dynamics simulation is executed.

The Featherstone's algorithm [67] is applied to our simulation model to solve a forward dynamics problem. Details are described in Appendix A. The generated torques defined by Eq.(3-16) and (3-18) (are added to Q_i in Eq.(Appendix 12) and (Appendix 15), and the external force exerted by a wind (Eq.(3-17)) is added to the gravity force in Eq.(Appendix 3) and (Appendix 7).

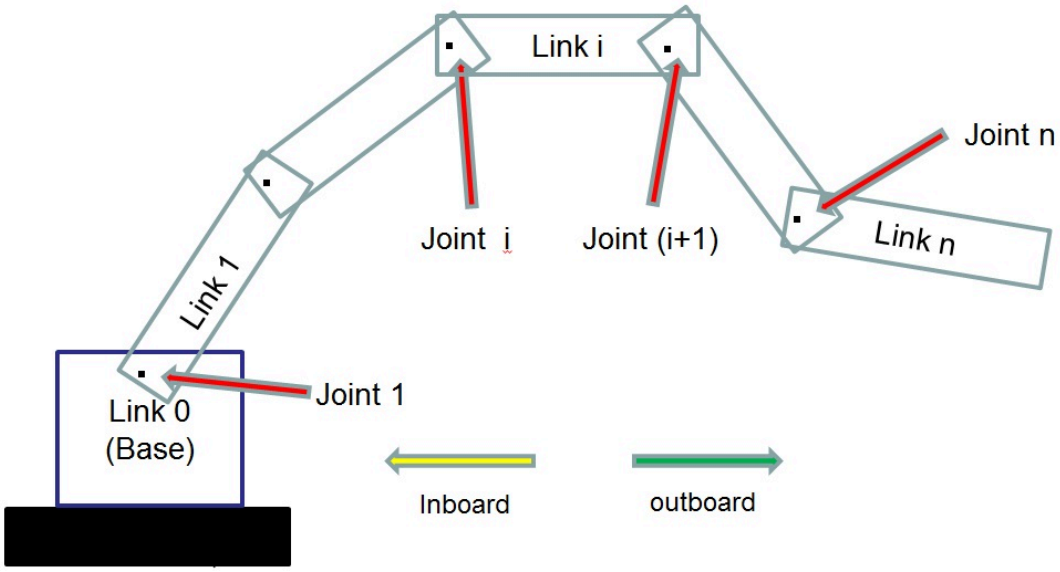

Figure 3-13, Link and Joint Indexing conventions for serial linkages

3.2.32 Constrained Forward Dynamics [20][41][60]

Consider serial links shown in Figure 3-13 by way of illustration. The links are numbered starting from *1* to *n*. Link *1* is attached to a fixed base (the "fixed base" is considered as link 0), and link *n* is the most lateral link. The joints are also numbered from *1* to *n*. For ($1 < i < n$), link *i* has both an inboard joint i and an outboard joint (i+1) and link *n* has only an inboard joint. Here inboard joints are closer to the fixed base, while the outboard joint is the other end of a link. When each joint only allows one degree of freedom, a compact parameterization of the configuration space of the links is the vector of joint positions $\mathbf{q} = (q_1, \ldots q_n)^T$. The following process is needed for solving this.

Given the positions **q** and velocity **q'** of the *n* joints of a serial linkage, the external forces acting on the linkage□ and the forces and torques being applied by the joint actuators. Then the resulting accelerations of the joints, **q"** is found.

In this system, there are 6 DOFs, which are x,y,z-translates, and x,y,z-rotations. In this case, we utilize just 4DOFs, since the target object (character's hair) is the 2D structure.

In the case of using 3D character's hair, compiling database [43] [44] [45] have become a common way of handling a corpus or scattered data. A strong point of the method is that this simulation is custom-designed by an animator for a specific target animation. All motions of hair strands in the simulation are ideally designed by an animator so as enable target hair motions to be extracted. The animator can even define the specific forces needed to obtain a motion that does not conform to general physical laws. The force animators can apply is an impulse function. Since the animator needs to design hair that is close to the sketched hair in the key frames, we consider the key-frame as the indicator of hair motion. For instance, to bend a hair strand dramatically, the animator should only apply a force to the middle part of the hair strand. Moreover, to make a motion in which only the hair tip moves, the animator should define forces applied only to the tip. Figure 3-14 and Figure 3-15 show the simulated situations such as wind blowing and the character vertically moving respectively. The black arrow represents the direction of force in each situation. This is defined based on the key-frame relationship. This model is explained more deeply in the Appendix of this section.

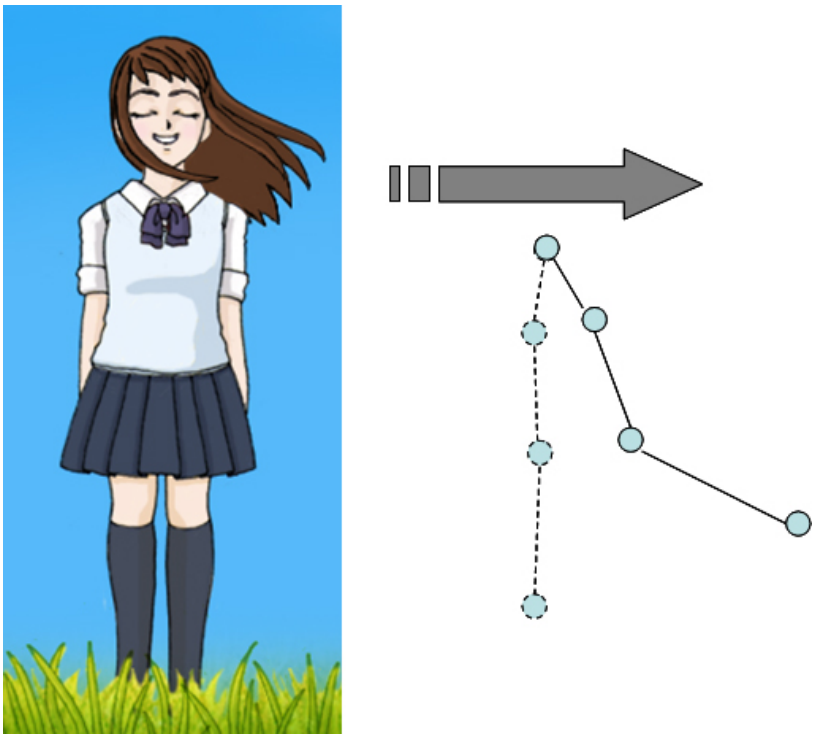

Figure 3-14, Wind Blowing

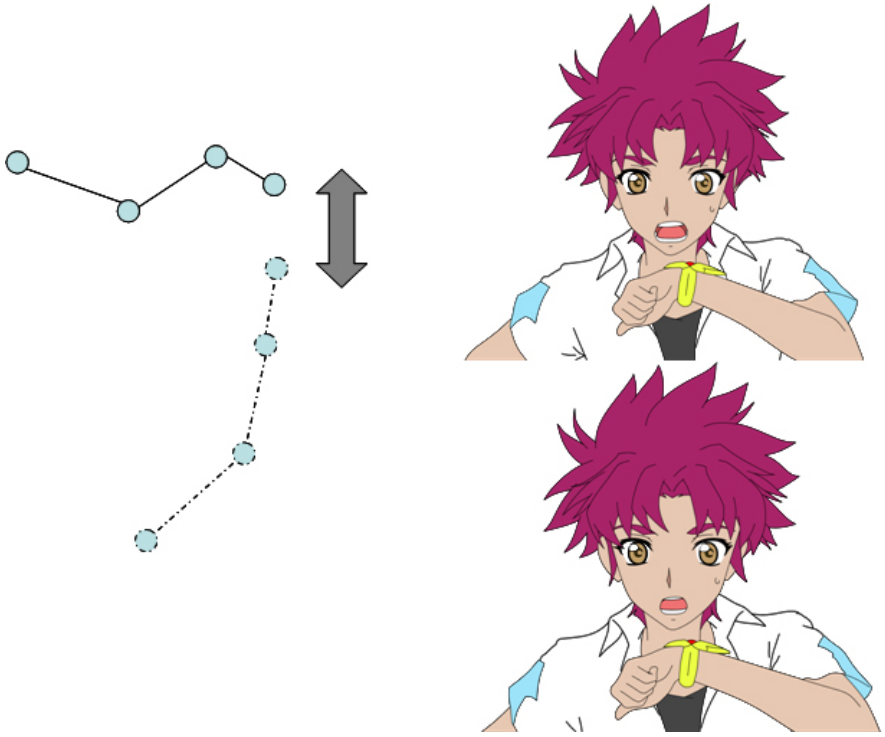

Figure 3-15, Character Moving Situation (vertically)

3.2.33 In-between Creation by applying Convergence Force

The objective of our paper is to create in-between frames based on our simulation model. To create in-between frames, animators have to draw two key frames. We denote the starting key frame f_0 and the target key frame f_1. As for in-betweening for hair animation, the hair motion itself would not converge on the target key frame without any constraint. Thus, we define a force called *"Convergence Force"* to the basic equation.

Two skeleton structure, S_0 (l_0, Q_0), S_1 (l_1, Q_1) are automatically constructed in each key frame respectively. A state of a skeleton S_i is defined by the lengths of each link l_i and the angles of each joint Q_i. We create in-between skeleton states from S_0 to S_1.We utilize the forward dynamics simulation to control the joint angels, and the lengths are simply interpolated from l_0 to l_1. Therefore, the objective target state is S_{obj} (l_0,Q_1) To converge the motion on the target state, we apply the joint torque toward the target angle at each joint when the position of the joint is close to the position at the target state. The torque is called the convergence torque at a joint i and defined as follows.

$$\tau_i^{conv} = K^{conv}(\theta_i^{obj} - \theta_i) \tag{3-19}$$

To make the resulting motion more naturally, the coefficient K^{conv} increases gradually. If the current angle of a joint is close enough to the target shape, the motion of the joint is forced to stop. In this case, the motion of the linkage might be unnatural because the joint velocity is suddenly set to zero when the condition is satisfied. Therefore, the certain in-between frames from the obtained in-between frames with the feature of the motion need to be selected to make the motion natural.

3.2.34 Feature Frame Selection [56]

There are two reasons why we need to apply the feature frame selection to the obtained in-between frames. One is the requirement for the animation. In the real production, the number of in-between frames is strictly specified in story-boarding. Since technologies in this paper are aiming for the real production, this process should also follow the industry specification. After the system obtains the in-between frames, animators have to choose the certain numbers of the frames. Because the simulation takes time to finish, the number of simulated frames is usually more than the animator's specified number. The other is that there might be a skeleton length difference between key frames. This rarely happens because there is not big difference in the shape and length between adjacent key frames. Thus the in-between-frame selection algorithm can be utilized as a Feature Frame Selection.

Since a curved line usually has discontinues part at connection points (see Figure 3-16, there is a step edge that is considered as a corner.), however, an extraction method for extremum does not work properly on a curved line of feature value directly. Therefore, both extremum and corners should be considered as key frames. When an isotropic Gaussian is utilized for creating an average curve, an unwanted key frame is selected where the average curve has only slight changes while an extreme right side is left out.

For reducing the artifact, a skewed Gaussian kernel is applied in place of the isotropic Gaussian kernel as a weighting function [56]. Kernels' shape is adjusted automatically to give low sampling density in a direction where the original curve does not change extensively. The process of adjusting for the shape is illustrated below.

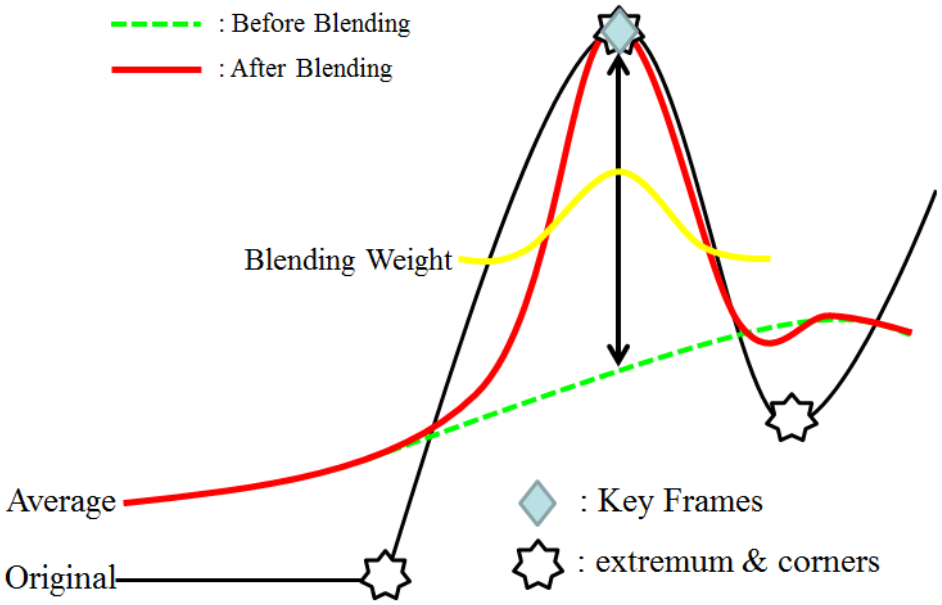

Figure 3-16, Average Curve Creating (this figure is quoted from reference [56])

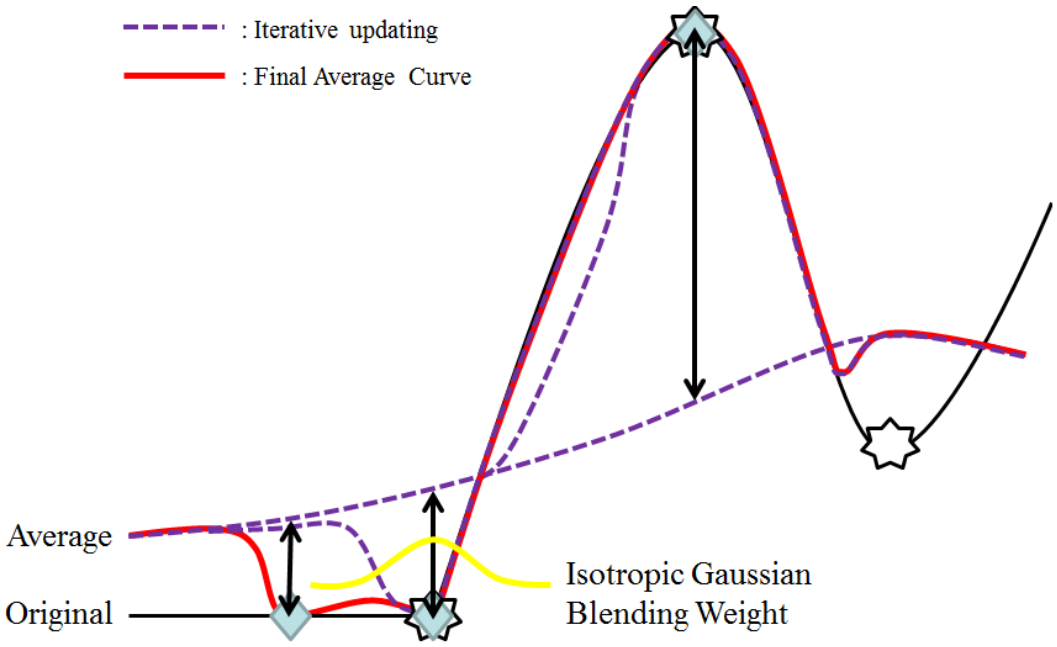

Figure 3-17, Artifacts due to Isotropic Gaussian Weighting
(this figure is quoted from reference [56])

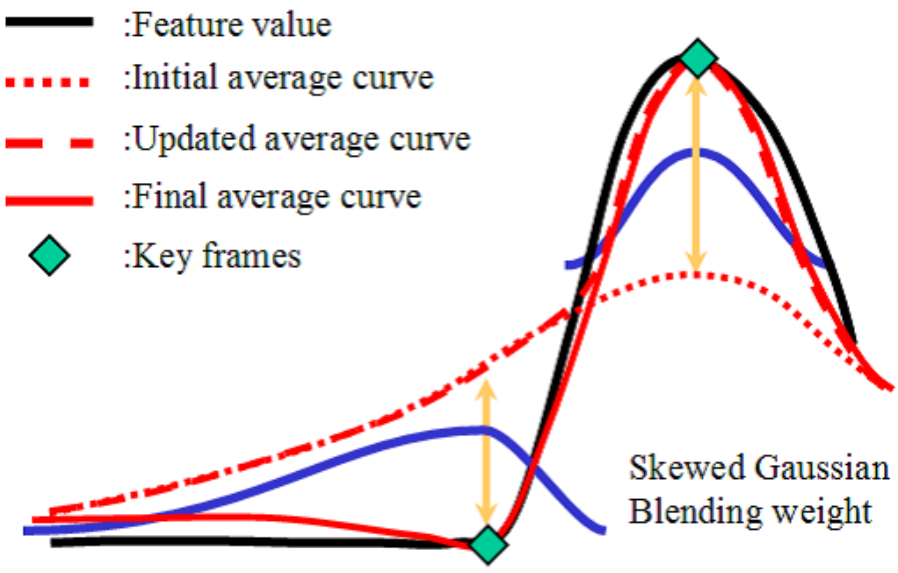

Figure 3-18, Average Curves Creating for Feature Frame Extraction
(this figure is quoted from reference [56])

First, we select key frames based on the algorithm proposed by Yasuda et al. [56]. Their algorithm has been developed based on the algorithm proposed by Assa et al. [83]. These methods extract local extrema of a curved line of the feature vectors. Figure 3-17 depicts the algorithm briefly. In our case, the joint angles, angular velocities, angular accelerations are utilized as feature vectors.

The local difference *y* between the average curve C' and the original curved line C around key-frame k is denoted by

$$y_i = \| C(i) - C'(i) \| / \| C(k) - C'(k) \| \tag{3.20}$$

Where, i is an adjacent frame of the keyframe *k*. the normalized residual errors are denote by y_i. By coordinating the blending weight proportional to the residual errors, the average curve is created to match the original function locally as shown in Figure 3-18. Then, the artifact around the corner edges is reduced. The process starts by finding a parameter σ of the Gaussian function, $\exp(-x^2/\sigma^2)$, that best matches the residual error distribution, minimizing the squared error *E:*

$$E = \sum_i (y_i - \exp(-\frac{x_i^2}{\sigma^2}))^2 \tag{3.21}$$

where x_i is an absolute value of k minus i. This minimization problem can be transformed into a linear regression problem that minimizes J:

$$J = (X - \sigma Z)^T (X - \sigma Z), (0 < y_i < 1), \tag{3.22}$$

where X is the column vector of xi, and Z is the column vector of $\sqrt{-\log y_i}$.

Here, y_k = 1 and y_i of adjacent frames take values between 0 to 1, and the neighbors with y_i = 0 are excluded for the fitting.

Using this in-between extraction, the system obtains the feature in-between sequence from the animated result. Animators can specify how many frames they need before starting the extraction. Based on the specified number, selected in-between frame is determined.

Figure 3-19 shows the in-between selection result. Four in-between frames with the features of the motion are automatically selected.

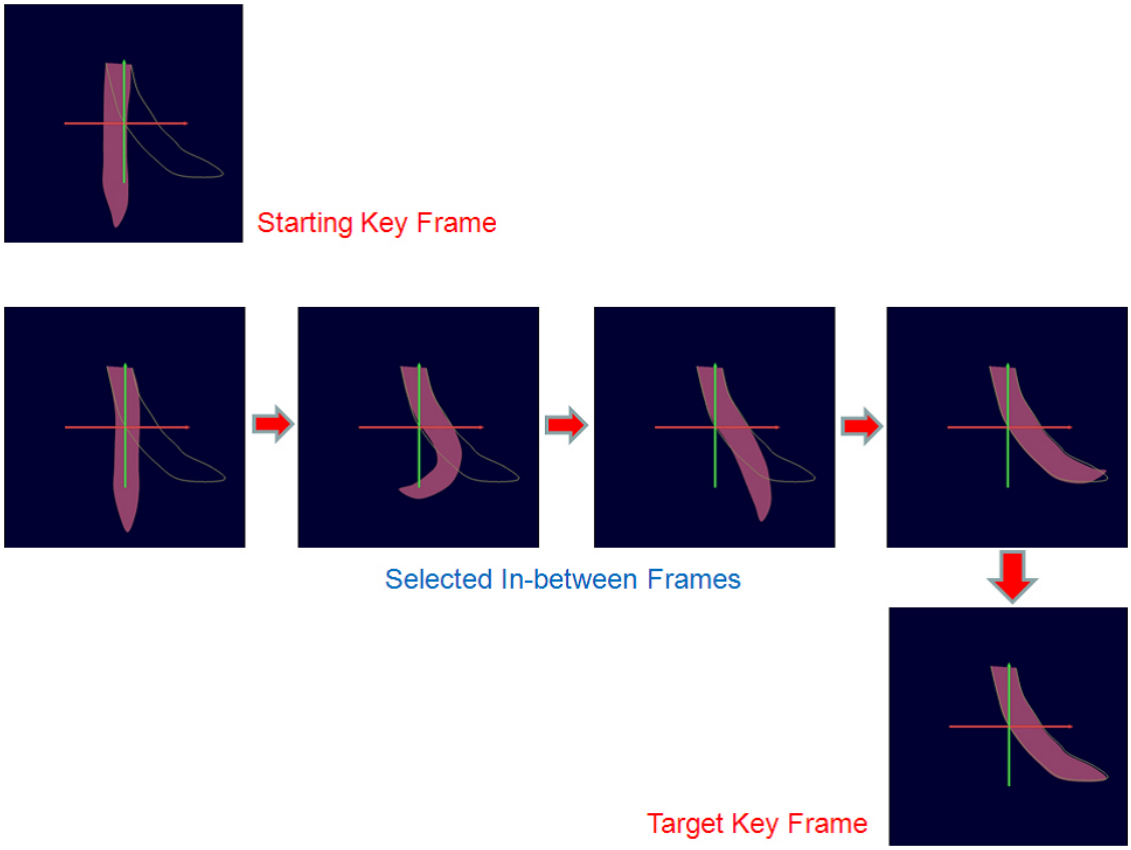

Figure 3-19, In-Between Selection Result for a Windblown Motion

Chapter 4: Stylized shading and highlighting, and Shadowing

4.1 Automatic Shading and Highlighting for 2D hand-drawing image

4.1.1 Introduction

Anime has its unique recognizable style of rendering and a wide range of artistic expression way. As a special art form, the visual styles of anime can vary from artist to artist or by studio to studio. But a commonality exists and is viewed as the characteristics of anime. Anime-like shading is one of the styles distinguishing it from other animations. As illustrated in Figure 4-1(a), the non-photorealistic highlight, shade and self-shadow are the essential elements of Anime-like shading, which are utilized to emphasize the lighting, depth, material and even the emotion of the character.

As shown in Figure 4-1(b), animators usually draw highlight and shade lines to create highlight, shade, self-shadow and shadow. To distinguish them clearly, we define a continuous surface as a surface with a uniform normal direction along the view direction at each point. Highlight and shade are the areas with higher or lower intensity compared with other areas in a continuous surface due to the surface reflectivity.

Thus, the rendering result in Anime tends to fill in the same color in one region, such as highlight region, shade region, and the original color region. In this sub-section, we are introducing an Anime-Like shading system which focuses on the highlighting, shading on 2D characters in Anime, which is suitable for Anime production pipeline.

4.1.2 Animators input and Overview of the approach

T the only input required is a hand-drawn character expressed by the combinations of strokes. First, we distribute the initial vectors on the strokes according to several conditions of the stroke such as open, closed, and connected or animator's preference. From the initial sparse vectors, an energy minimization with the divergence free constraints is proposed for estimating vectors for the whole character. Then, based on the estimated vectors, the toon-shading technique is applied to render the character to achieve anime-like shading and highlighting. Several ways to intuitively edit the rendering results are provided to reflect animator's preference. Animator-oriented shading and highlighting are thus achieved semi-automatically with minor user-interactions. Figure 4-2 and Table 4-1 represent the process of the approach respectively.

As demonstrated more practically in Figure 4-2, the image is drawn by animators first, and then the normal vectors on the surface are estimated (normal vectors described as blue arrows). Finally, the image is by toon-shader based on the light source position. After animators obtain the rendering result, they can edit the result, in this figure 4-2, they add the crescent moon shape of highlight.

Figure 4-1, Example of Shading in Anime
(a) Shading for Anime, (b) highlighting and shading lines
(Copyright by Anime International Co. (AIC), Inc.)

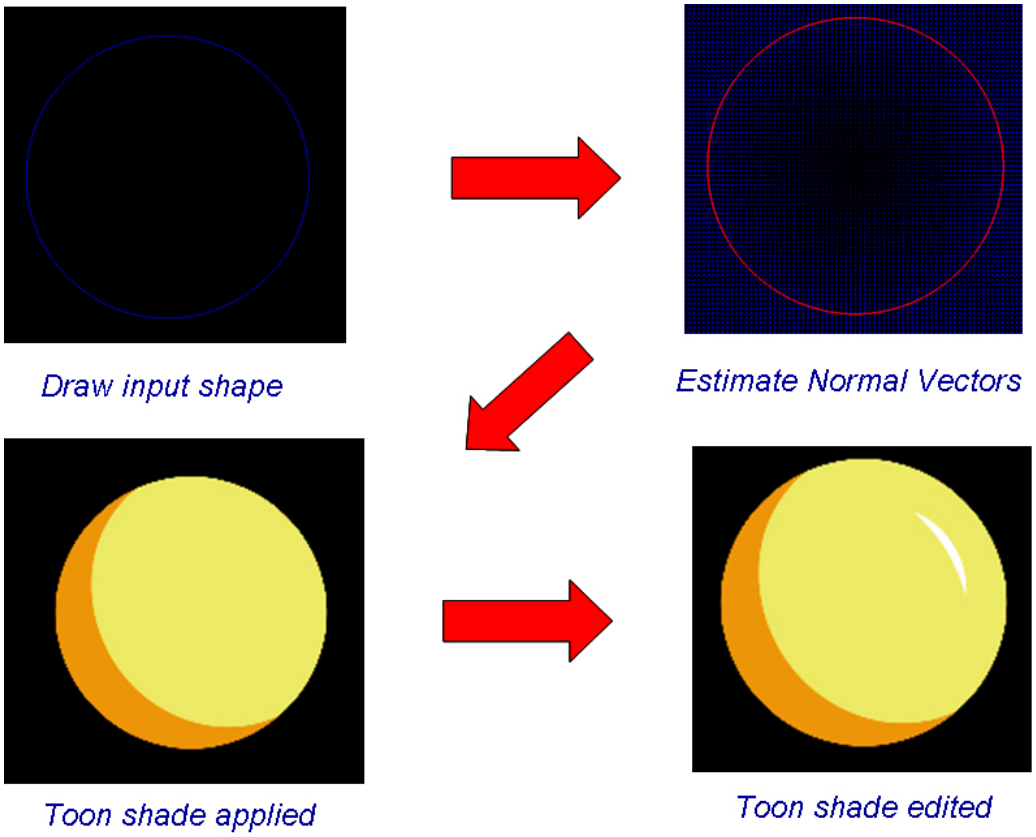

Figure 4-2, Process Overview for Rendering in the method

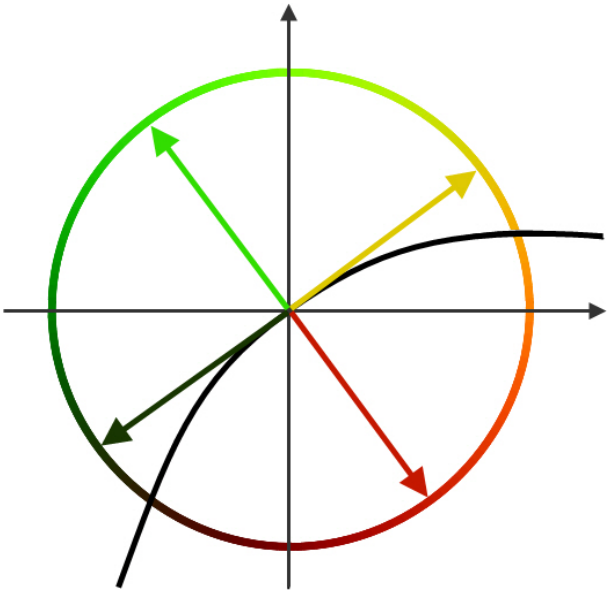

Figure 4-3, Color on the circle represents a color map for indicating direction of vectors (Curve shown in black is an input stroke)

4.1.3 Surface Normal Vectors' Estimation

4.1.31 Initial Vector Assignment

To estimate 3D surface normals for an input 2D image, Tai-PangWu et al. [79] proposed a novel idea by using a reference models (Shape Palettes). The normal specification method is quite convenient and straightforward, but might not be intuitive enough for 2D animators. Based on our investigation, several 2D animators encountered difficulty in handling the mapping from strokes located on a 2D image to corresponding Shape Palette in order to accurately represent what animators want in their minds.

In Lumo [51], visible edges which form the exterior silhouette and interior folds, are viewed as being composed by edge points where the surface normal is perpendicular to the eye vector. Normal vectors are automatically generated along the points and consequently interpolated for the whole character. The interpolated surface normals based on this assumption forms curved surface with limited curvature.

In our method, vectors are set along the strokes with the same or perpendicular direction of stroke tangents, as indicated in Figure 4-3. The direction of vector indicates the height of surface is decreasing along the same direction and the magnitude of vector is introduced to control the rate of change of surface height. That is, if the magnitude of vectors is large, it generates a large rate of changes in surface height and the surface becomes stiff. Figure 4-4 shows examples with different settings of vector on the middle vertical line stoke and the magnitudes of vectors on the rectangle equal to 0. From these examples, 2D animators can easily understand the vector assignment scheme and apply it to more complicated drawings.

As indicated by Cole et al. [84], 2D lines drawn by artists are usually related to the local geometry properties of 3D surfaces such as silhouette lines, occluding contours, ridges, valleys, and diffuse shading. In 2D animation drawings, especially Japanese anime, the majority of drawn strokes can be viewed as occluding contours. Based on this observation, the initial vectors are automatically assigned with the direction perpendicular to the stroke tangents. In Smooth Sketch [71], cusps and T-junctions, as endpoints of hidden parts of the occluded contour, are correctly completed in a right topology. In our method, the magnitudes for vectors near the T-junctions along the visible part of the occluded contour are automatically reduced to avoid affecting the vectors specified for the contour occluding it; see Figure 4-7.

Initial vectors are automatically assigned in Figure 4-5(Left). 2D animators may further adjust the directions and magnitudes of vectors easily with simple mouse clicks, referring to the vector assignment scheme. Besides the drawn strokes, some guide

strokes are optionally added to initialize vectors in desirable positions and directions. Figure 4-5(b) shows an example of the initial vectors adjusted by 2D animators. To give the artists as more flexibility in controlling the shape surface, more advanced options are provided such as the magnitude patterns and vector rotation angles, as depicted in Figure 4-6.

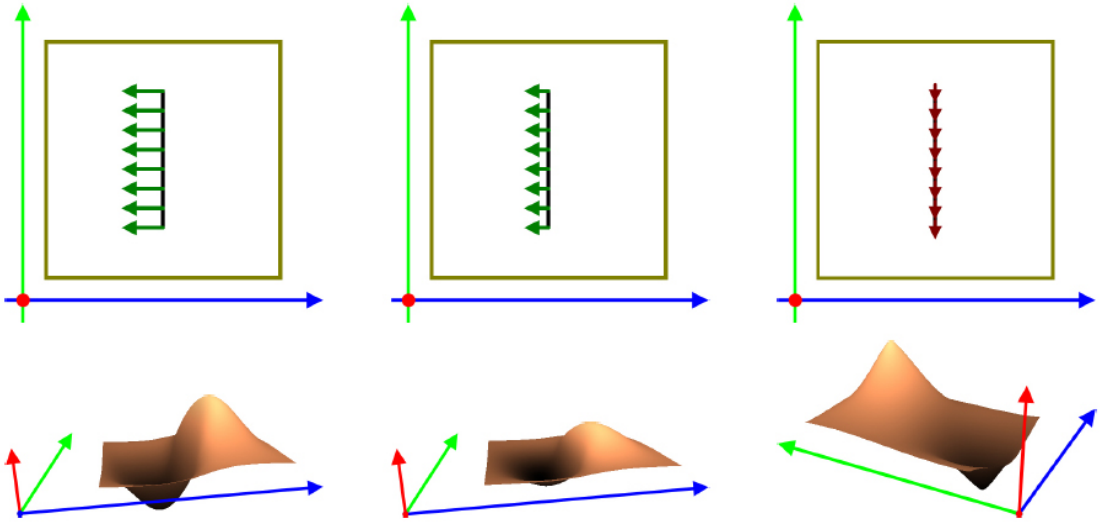

Figure 4-4, Representation for changes of direction or magnitude of initial vectors affect the shape of surfaces.

Figure 4-5, Vector Assignment
(Left) Auto-Assigned, (Right) User-Adjusted

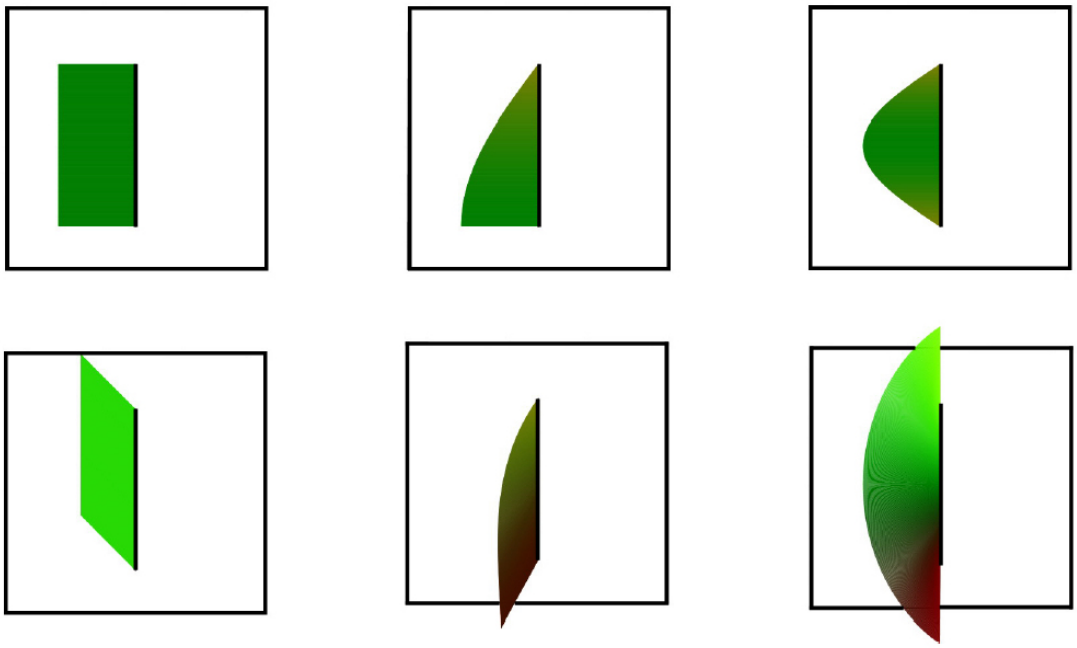

Figure 4-6, Different Settings of Initial Vectors on the Straight Line Stroke:
These are obtained by simple scaling and rotation operators from the vectors on top left figure.

4.1.32 Vector inpainting algorithm

In this section, the proposed vector inpainting method is explained and a fundamental difference with the previous methods for estimating dense vectors from sparse vectors [79], [80] are described.

Let $\subset \mathbf{R}^2$ be a rectangular domain, $R \subset \Omega$ be a closed region of given characters, and Γ be a set of open strokes in R and the boundary contour of R. These strokes usually make a shape of characters to look like 2D drawing as 3D surface. After piecewise smooth vectors $n^* = (n_1^*, n_2^*)$ on Γ are initially assigned, the previous models are basically to obtain or modify dense vectors from minimizing energy functional:

$$\min_n \int_\Omega |\nabla n|^2 + \frac{\eta}{2}\int_\Gamma |n - n^*|^2$$
$$|\nabla n|^2 = \sum_{i,j} (\partial_{x_i} n_j)^2 \tag{4.1}$$

where η is the positive number. Note that we will explain how to set and control the initial vectors in the next section. The first term is the H^1-regularization term and the second term is the L^1-fidelity term. Note that the model in [79] has an additional term of minimizing the surface curvature. The estimated vectors have been successfully utilized in surface reconstruction. It is obvious to generate better surfaces if the estimated vectors n from n^* approximates the projected surface normal vectors onto $\mathbf{R}^2$, because the vectors satisfies the integrability condition $\nabla \times n = 0$ [85], [86]. However, it is still a challenging problem to estimate such a normal vector field based on the functional in (4.1), especially where any further information such as a height field or a discontinuity map are unknown [81], [87].

Inspired by an equivalent integrability condition in the vector field $t = (n_2, -n_1)$ which is orthogonal to $n = (n_1, n_2)$ [75], we propose a minimization based on H^1n with the divergence free constraint:

$$\min_{\nabla \cdot t = 0} \int_\Omega |\nabla t|^2 + \frac{\eta}{2}\int_\Gamma |t - t^*|^2$$
$$t^* = (n_2^*, -n_1^*) \tag{4.2}$$

In image inpainting, the technique in [74] firstly utilized an analogy to incompressible Newtonian fluids in order to propagate smoothness measure along the isocontours of surface. Later, two-steps algorithm [75] was introduced in image inpainting to enforce the incompressibility condition, $\nabla \cdot t = 0$, for estimating tangent vector field in the inpainting domain based on the total variation functional. The divergence free constraint in the proposed equation (4.2) makes a clear difference from the previous equation (4.1).We incorporate with an equivalent integrability condition in the estimation of dense vector fields. For the surface reconstruction, it allows to utilize a standard minimization with H^1n for the difference between the surface gradient and the estimated vectors. Moreover, it is natural to have more faithful results according to the changes of an initial sparse vector field.

In Figures 4-7 and 4-8, noticeable differences are shown in the orientation map and the shape of reconstructed surface. The orientation map indicates that the direction of inpainted vectors and the color represents the direction according to Figure 4-3. The previous equation (4.1) is a harmonic extension from the initial data and it looks a linear extension of the orientation of initial vectors. The model has not only a harmonic extension but also a term of enforcing the integrability constraint. This constraint yields more sensitive orientation changes in the domain and generates more plausible surfaces according to the changes of initial vectors. Comparing with many initial strokes in [81] for Figure 4-8, we use less number of strokes to make better result. For toon-shading results in Figure 4-9, the difference comes from the different shape of surface. We numerically solve the proposed model by using the augmented Lagrangian method [88].

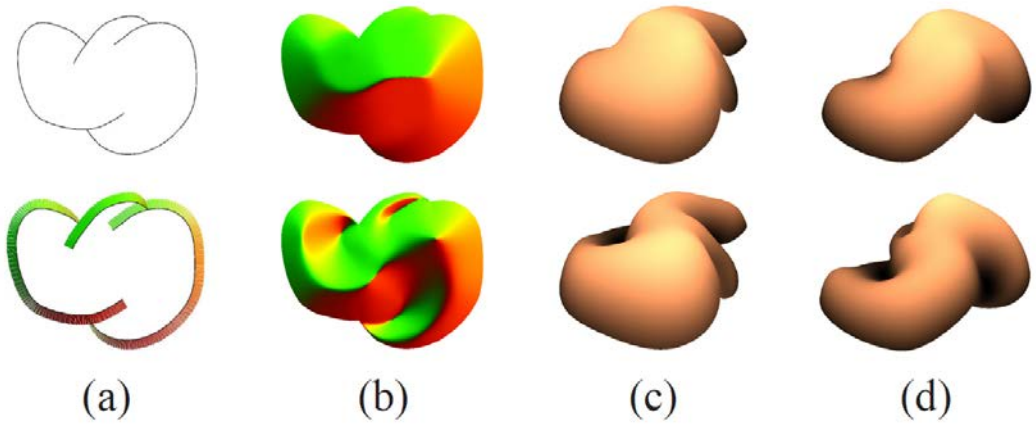

Figure 4-7, Inpainting Result I, (a) is initial strokes and vectors.
(b) is the orientation map of inpainted vectors. (c) and (d) are results for comparison between the equation (4.1) (top) and the equation (4.2) (bottom).

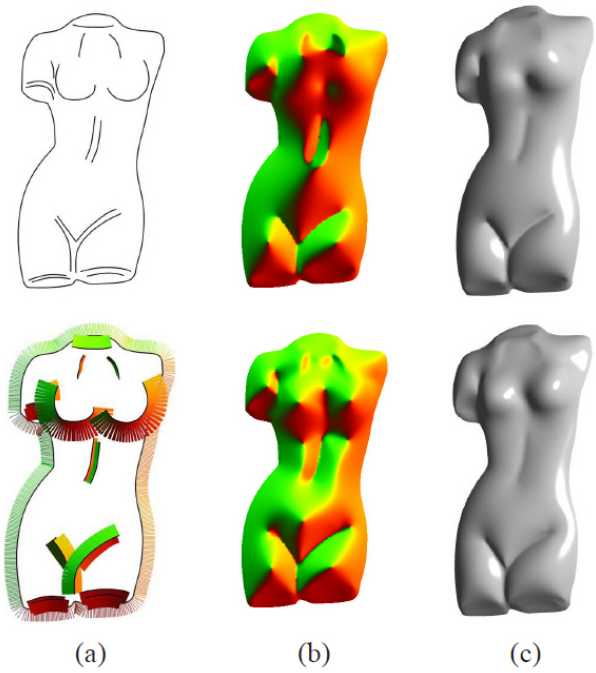

Figure 4-8, Inpainting Result II
(a) is initial strokes and vectors.
(b) is the orientation map of inpainted vectors.
(c) is results for comparison between the equation (4.1) (top)
and the equation (4.2) (bottom).

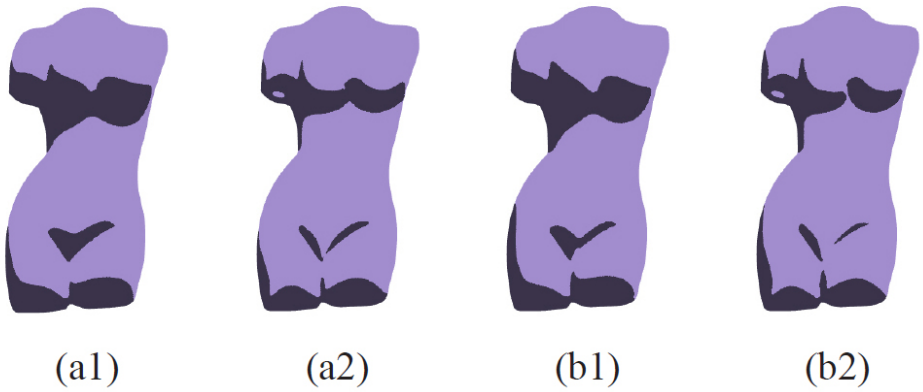

Figure 4-9, Inpainting Result III
(a1) and (b1) are toon-shading results from (1).
(a2) and (b2) are toon-shading results from (2).
(a1) and (a2) have exactly same light condition.
Also, the light in (a2) and (b2) is synchronized.

4.1.4 Shading Pipeline

In this section, the shading pipeline to obtain anime-like shading result is described. As described in Section 4.1.3, surface normals have been obtained, the shading is applied to the input image based on the obtained surface normals.

4.1.41 Toon-Shading

Toon shading has been a well-known method to render the target model for more than 10 years. It can be seen in a variety of 3D renderers, game industries, entertainment systems, and animations. The idea of the shading is quite simple but powerful, which extends Lambertian shading model by using dot products between normals and light vectors, normals and eye vectors at each pixel to index into a texture that represents how the shading is implemented (Figure 4-10). The result can be “highlight”, “shade”, basic part color, and more colors depending on the texture.

In our case, to obtain the anima-like shade, we prepare the 4 colors texture that represents “highlight”, “shade”, basic part color, and the contour lines (usually black, shown in Figure 4-11). Based on the obtained surface normals, we can apply the toon-shading to the input image itself even though we do not have a 3D structure of the input drawing. First, we need to obtain dot product s between a normal vector and the light vector and another dot vector t between a normal vector and the eye vector at the target vertex. Then, *s* is set to x-axis and t is set to y-axis. These axes are correspondence to texture coordinate. Thus, based on these dot product values, the color on the target vertex is determined according to the result on the texture color.

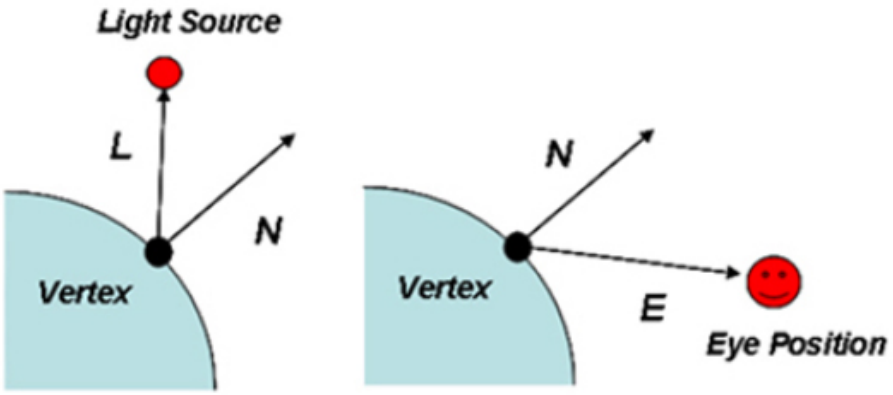

Figure 4-10, Overview of Toon Shading

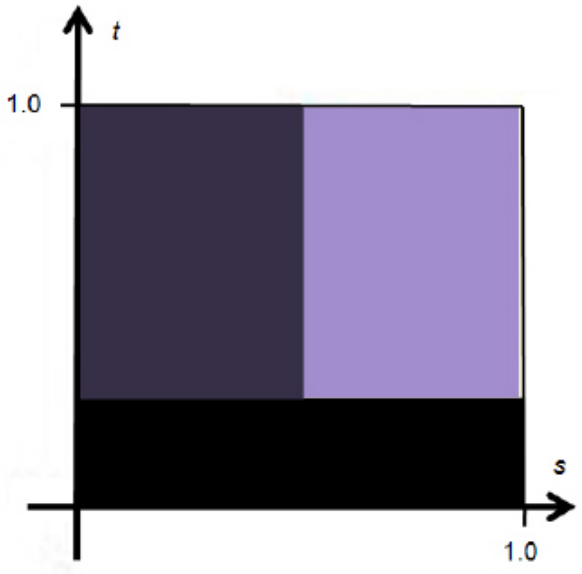

Figure 4-11, Example of Texture for Toon Shading in this case
Black: Contour Line, Dark Purple: Shade Part, Purple: Base Color, White: Highlight

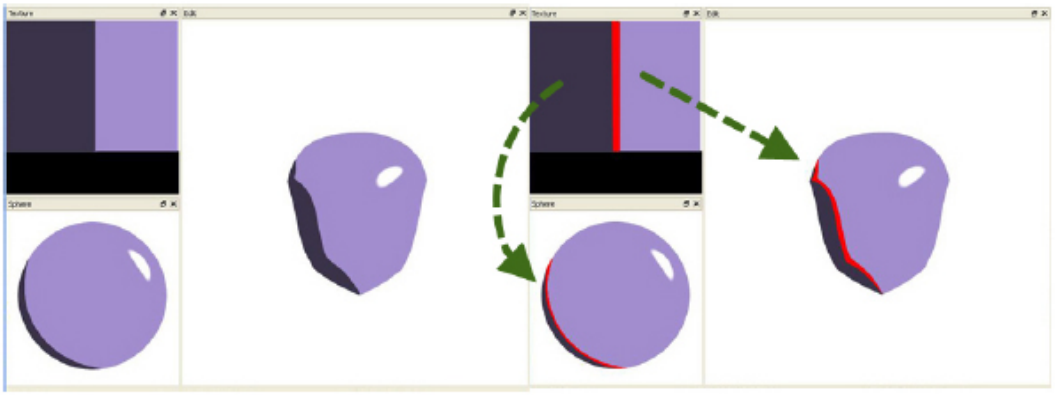

Figure 4-12, Texture Color Editing

4.1.42 Editing Operations

After the rendering result is obtained, animators may want to edit the obtained result a bit to reflect their artistic features and preferences. Thus we provide animators with several ways of intuitive editing system. To see the several kinds of shading result we also give 3D control (translation, zooming and height adjustment) to animators.

Texture Color Edit

Animators can edit a texture directly put the color onto the texture. Figure 4-12 shows the example for the texture color editing. When the mouse is pressed and moved on the texture, the correspondence pixels (a cluster of pixels) are filled by a chosen color. After filling the color, the shading result will be changed based on the updated texture. This operation is suitable for the boundary color editing. In a similar way, animators edit on the Toon Shading Sphere model and reflect to the texture and the actual rendering results in real time.

Obtained Result Edit Directly

Animators can edit the obtained result rendered by our shading pipeline. This is more straightforward way to edit the obtained result and the edited color information reflects back to the texture and the Sphere model (Figure 4-13).

Light Source Controls

Lighting Adjustment

- For directional light: change the direction of the lighting vector.
- For point light and spot light, change the position of the light.

Spot light rendering with cut off adjustment

- For spot light rendering, in the Toon shading shader: Named the vector from the origin to the light position V_1 and the vector form the each vertex to the light position V_2. If the angel between V_1 and V_2 is small than the cutoff of the spot light, the texture coordinates s will be calculated, else s will be set to 0.

Virtual Multiple lighting sources

- For multiple directional lights: each light will be assigned a weight, and the average among all the lights based on the weight will be utilized in the shader to obtain the rendering results.
- For multiple spot lights: for each spot light, s will be calculated, and the largest s will be utilized as the actual texture coordinates to render.

Vectorized the Obtained Result

To edit the obtained result globally, we apply a vectorization method [89] to the obtained result. First we need to extract contour and boundary lines from the obtained result by applying Laplacian Filter. Then, we apply the vectorization method to the

extracted lines. The method can assign the strokes onto the extracted lines. Then animators can edit the strokes to change the color regions globally. Finally, the desired colors are filled in the regions. Animators can thus reflect their taste and preference. Figure 4-14 is the procedures for this editing.

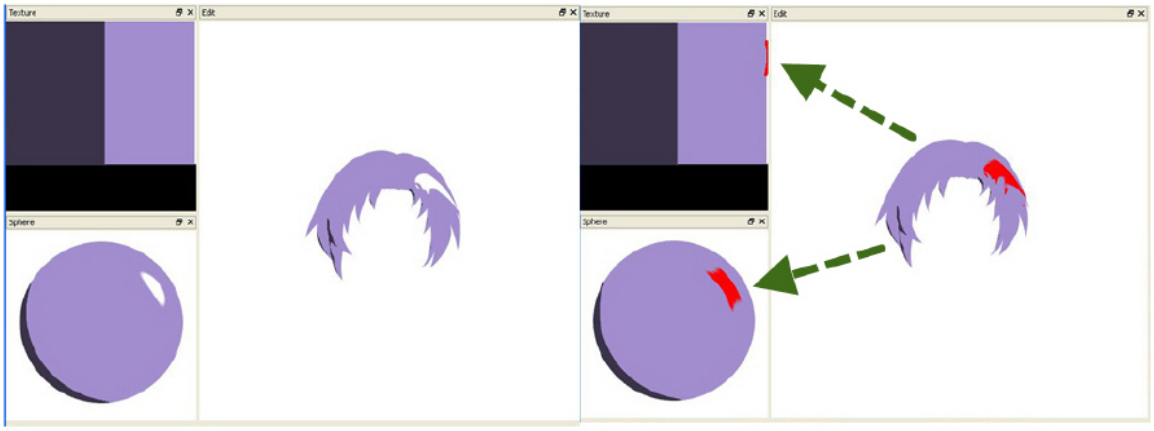

Figure 4-13, Directly Editing Obtained Result

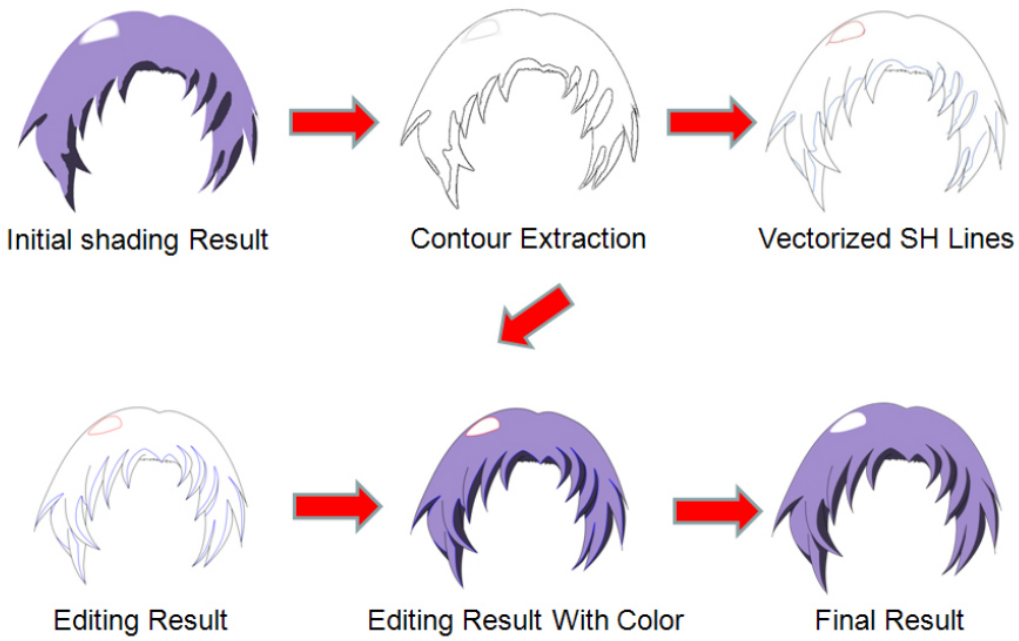

Figure 4-14, Editing Result: Vectorized Shading and Highlighting Lines for Obtained Result

4.1.5 Results and Analysis

In this section, we now demonstrate the results created through our shading pipeline. Since shading for hair is one of the most important elements to represent anime-like factors and very hard to obtain the appropriate shading result by previous methods, we start with the hair shading result.

Anime-like shading has the feature of sharp edges especially for hair shade. To obtain such a shading effect, skeleton of each hair strand is automatically detected based on the both hair strands, and tangent vectors are assigned along the detected skeleton accordingly. Figure 4-15(a) and Figure 4-15(b) show the skeletons and vectors generated automatically. Based on the vector setting, anime-like shading effect is achieved as depicted in Figure 4-15(c) with sharp shade edges. The result without the skeletons under the same lighting environment is shown in Figure 4-15(c) for comparison. These images show our initial vector assignment successful creates the sharp edges on the hair. Although we give an option to animators to assign the initial vectors' direction and amplitude, our automatic initial vector assignment has an enough quality to be utilized for shading.

Figure 4-16 shows the whole pipeline for our shading. First our method requires animator's hand-drawn image as an input. Next, the (semi-)automatic initial vector assignment is applied onto the drawn strokes. To obtain the surface normals, the vector inpainting is applied to the surface of the drawn character in each layer based on the initial vector sets. To achieve the anime-like shading result, we render the image by using toon-shading algorithm. The result we obtained at this point already seems decent. However, since our method aims to develop for animators, we provide them with intuitive editing operations. Figure 4-16(d) shows the edited result by an animator. As seen in this Figure our shading pipeline and editing system have successfully developed to create anime-like shading result with less animators' interaction.

Figure 4-17 shows the result created by two animators. They created these results less than a few minutes. Giving them a freedom, one created mild edge on the shaded region (Figure 4-17(b)). The other created sharper edge shade result (Figure 4-17(c)). He also changed the hair color to reflect his taste. It shows our pipeline can support animators to reflect their taste and preference easily, which animators desire to use.

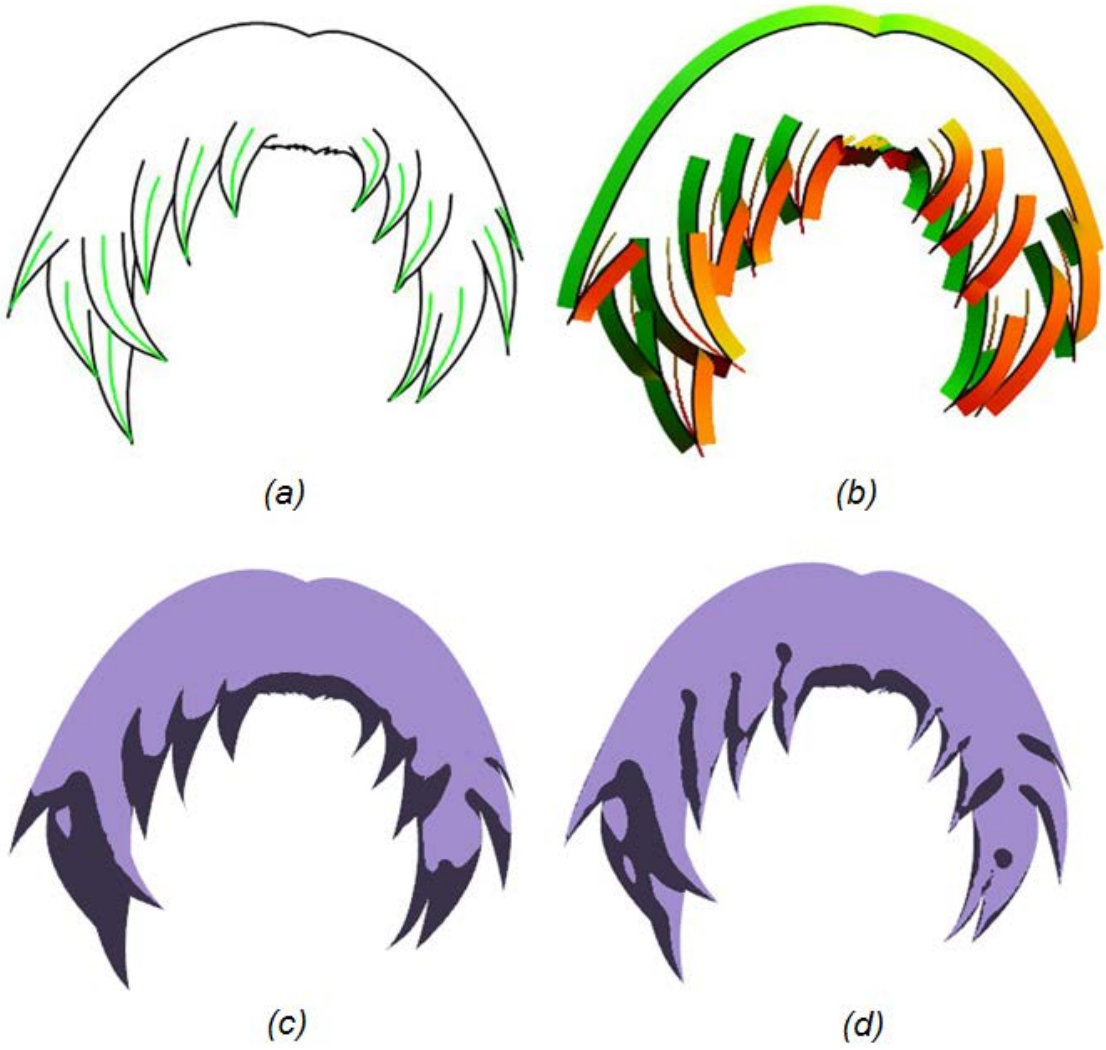

Figure 4-15, Hair Shading Result
(a) Hairstyle outlines with the detected skeleton
(b) Assigned initial vectors on the hairstyle outlines and skeletons
(c) Result with skeletons, (d) Result without skeletons.

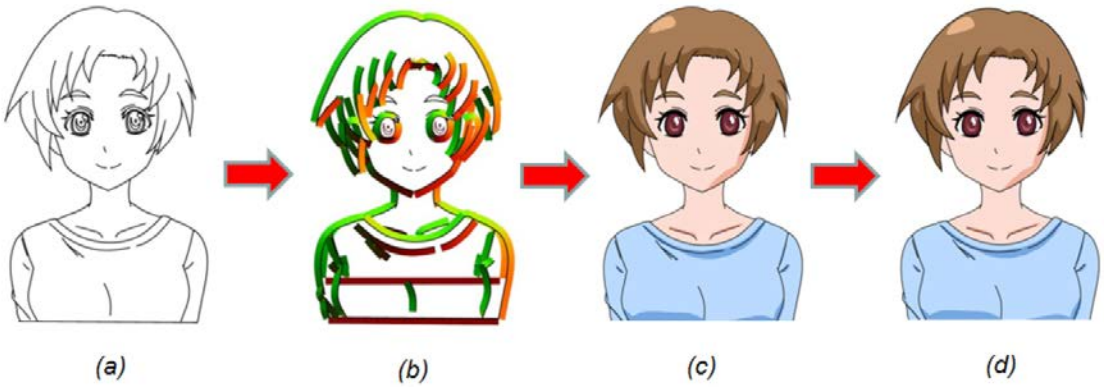

Figure 4-16, Review of our Shading Pipeline
(a) Input hand-drawing, (b) assigned initial vectors
(c) Processed Shading Result, (d) Minor Editing applied to (c)

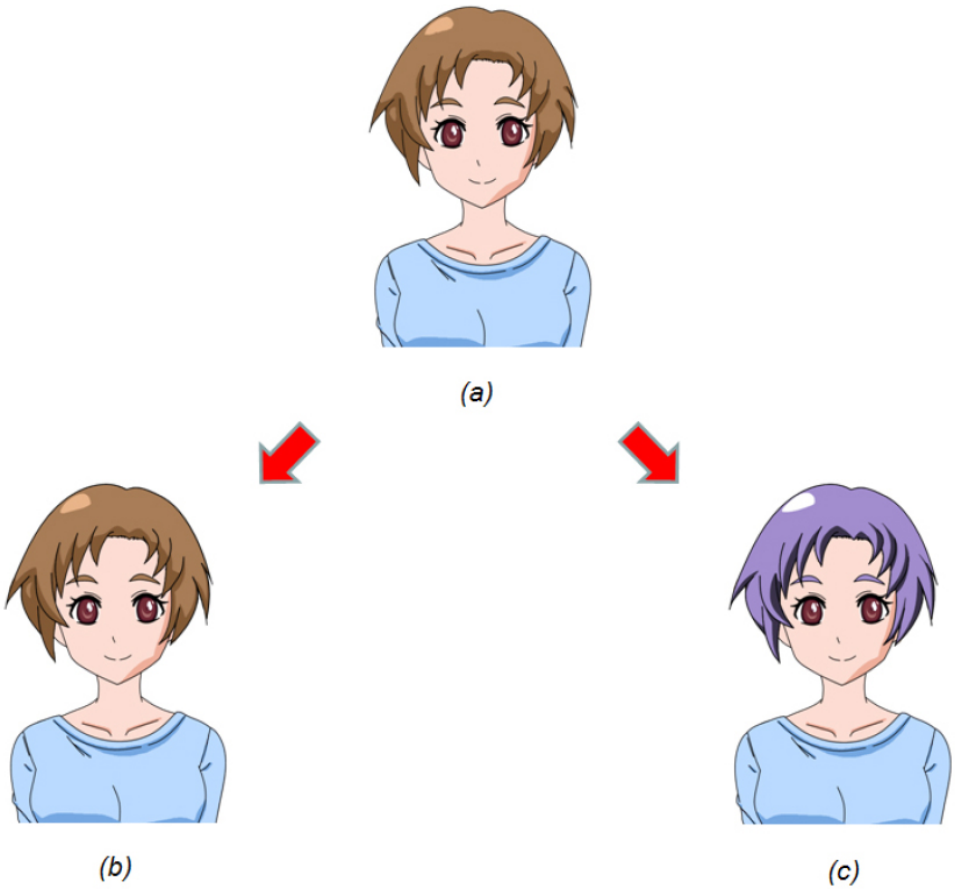

Figure 4-17, Various Results Reflecting Animators' Preference
(a) Processed Shading Result (b) Animator's Edit 1 (c) Animator's Edit 2

4.2 Interactive and Intuitive Shadowing for Anime

4.2.1 Introduction

In Anime, shadows are rendered not only to depict the relationship between characters or objects and background but also to grab the audience's attention to certain characters and scenes. Despite the importance and usefulness of shadows, animators might not draw shadows at all because of the lack of animators and time constraints in Anime production. Thus, the shadows in Anime play an important role in specifying a character's position. In fact, audience is able to perceive whether a character is standing on the ground or not from the position of shadows. Figure 4-18 shows the example of the role of Shadow. In this section, we describe our approach for rendering shadow instantly by using 2D character or object image sequence drawn by an animator. First, an animator is required to set character or object animation sequence with transparent information as input. Then, we apply Shadow Map to the input animation sequence for rendering shadow.

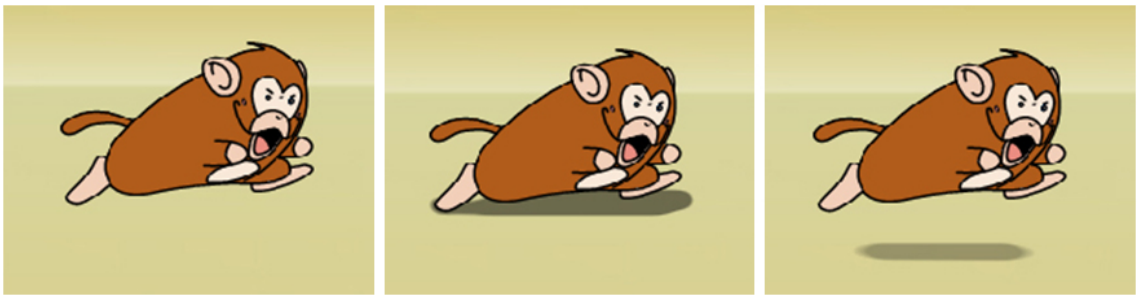

Figure 4-18, Role of Shadow, Specifying Character's Position:
(i) Uncertain position, (ii) Running, (iii) Jumping; in the left image, character's position is hard to judge without a shadow. However, the other two images enable audience to perceive the character's position clearly. Shadows are rendered by our approach.

4.2.2 Shadowing Approach for 2D Character and Object Animation Sequence

4.2.21 Input: Character or Object Animation Sequence

Animators utilize a character or an object animation sequence with alpha value as input in our approach. The advantage of using alpha value as a target of shadowing is that it is clear to distinguish whether a character or an object exists in the input animation sequence. The region where the alpha value exists (see Figure 4-19 as a reference) is utilized for shadowing. Thus, our approach stores the drawn character sequence with alpha value internally. Usually in Anime production, everything within these image layers is transparent apart from the characters or objects themselves for achieving layering for composition. In fact, all layers have transparent information. In this way, our approach handles alpha value to apply Shadow Map.

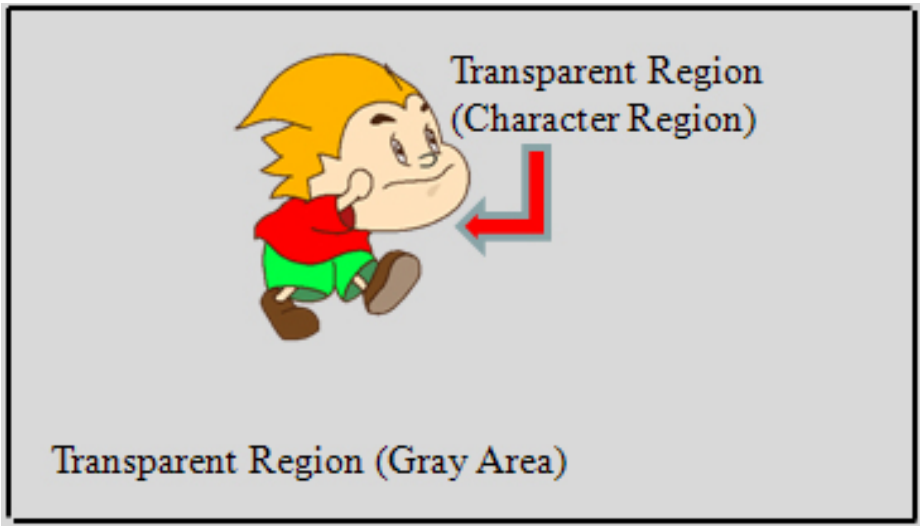

Figure 4-19, Example of the Input Character Sequence:
The character region is opaque and the rest of the image region (gray area) is transparent. We use such an animation sequence as input.

4.2.22 Apply Shadow Map to the Input Animation Sequence

Although there are several possible shadowing techniques that could be applied to our approach, we choose Shadow Map because of its interactivity and easiness for implementation. It is commonly utilized for rendering shadows [28]. Several research projects have been dedicated to developing a means of producing high-quality shadows in a shorter time [47].

While Shadow Map is usually applied to 3D objects, we applies Shadow Map to the character or object region in the input animation sequence. First, we obtain the distance from the light to an input animation sequence and the plane (virtual ground) of shadow projection in the depth buffer looking from the light position. A texture visualized from this stored information is called the Shadow Map. Then, looking from an initial view point, shadows are finally rendered onto the display plane by referring to the Shadow Map.

1. When we render the scene in figure 4-20 from the eye position, we need to check whether the reference point on the object surface is in the shadow region or not. If the point is inside of the shadow region, it is rendered as a shadow, if not, vice versa. To achieve this process, we need to obtain the position of the point in the world coordinate. This point is considered as the texture coordinate.
2. Next, the eye position is virtually set at the same position of the light source. Looking from the light source position to the reference position, we judge whether the reference point can be seen from the light source or not. If the point can be seen from there, the point is considered as the bright region. If not, the point is the shadow region. Thus, this process is same as a hidden surface removal process.
3. Here, before rendering the scene from the actual eye position, we need to store depth info from the light source and then send it to texture memory.
4. Then, as same as the normal texture mapping, the reference point of the texture coordinate is sampled. We compare the sampled Z-value and sent texture Z-value. If the texture Z-value is smaller than the sampled Z-value, the reference point is considered as the bright region.

In our case, the target object is the 2D image and the applied region is defined by the alpha value of the image. So we can directly apply the Shadow Map to the 2D character animation sequence.

4.2.23 Setup for the Input Animation Sequence, View Point, and Virtual Ground

Shadows are rendered onto the virtual ground using input sequence with transparent information. In fact, transparency existence or nonexistence in the input animation sequence can be utilized for making a decision whether to apply Shadow Map. Figure 4-21 shows setup for the input animation sequence, view point (eye position) and virtual ground respectively. The input animation sequence is set vertically to the direction from the view point. Then the level of the virtual ground which is invisible to animators is set at the bottom of the character's alpha value as the initial level. The shadow is cast onto the virtual ground depending on the position of the light source. Then the rendered shadow is projected onto the display plane.

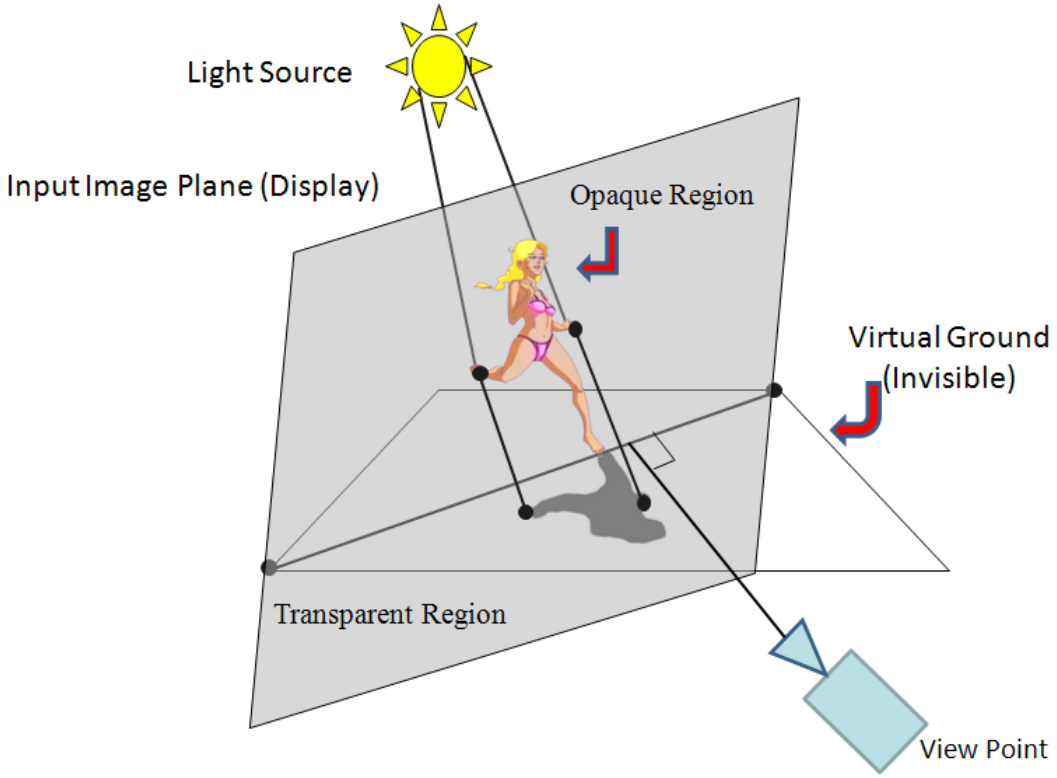

Figure 4-20, How to apply Shadow Map in this system

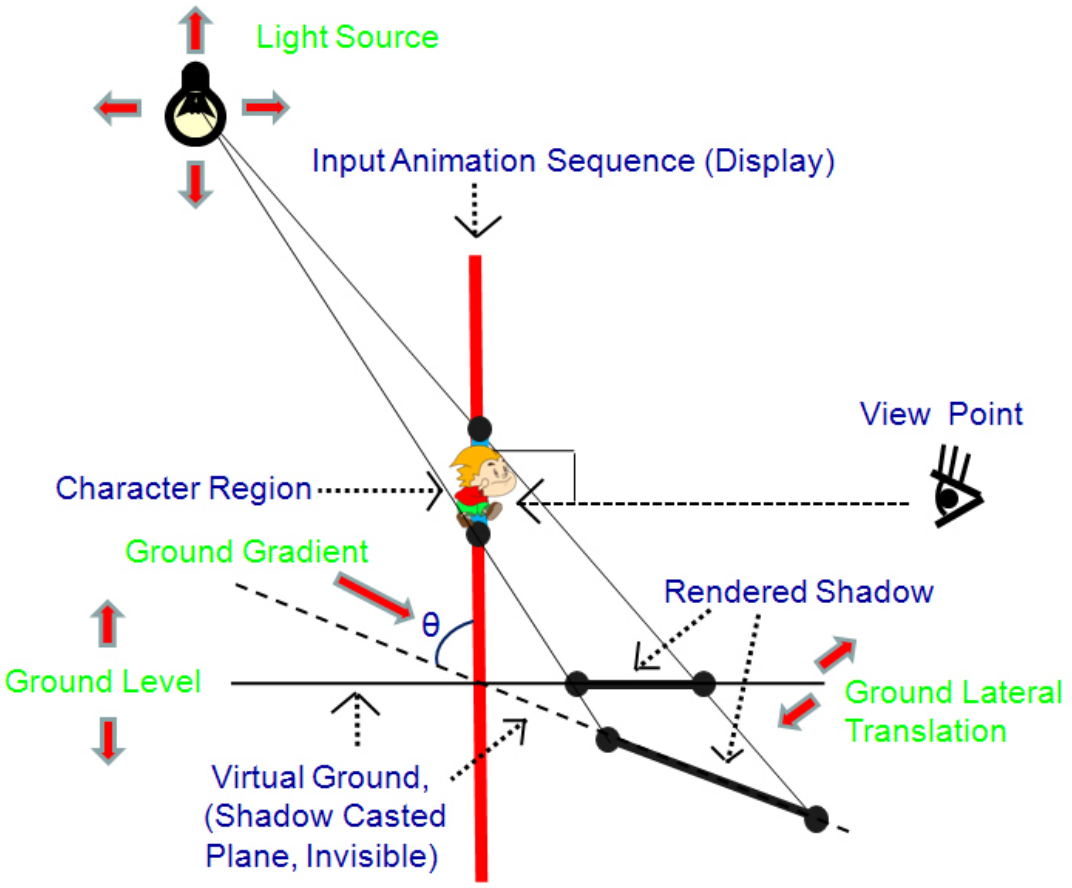

Figure 4-21, Apply Shadow Map to Input Animation Sequence (Side View): Input animation sequence is set vertically from the view point (camera). The initial level of the virtual ground is set as the lowest position of character or object. Gradient is defined as the angle between the input animation sequence and the virtual ground.

4.2.3 Editing Operations

4.2.31 Overview

After shadows are generated onto the virtual ground, our approach allows animators to edit the rendered shadow by adjusting parameters with several intuitive operations. As a result, animator-oriented shadow expression such as Anime-like shadow is easily and quickly achieved from the input animation sequence. The editing procedure consists of two steps. The first step is for the shadow position and shape editing, considered as the global editing. After setting the shadow position and rough shape, our approach allows animators to edit shadow shape more particularly such as blurring and simplifying.

4.2.32 Position and Shape Editing

Several parameters are utilized for editing shadow position and shape such as “Light Source Position”, “Ground Level and Lateral Translation”, and “Ground Gradient” as illustrated in red arrows in Figure 4-21.

Light Source Position: Animators are required to set the light source position first. The basic direction and scale of shadow is depending on this parameter. Animators can control the position of light source by the controller which is shown in the upper left circle in each image of Figure 4-22. Black region corresponds to the shadow casting direction. Therefore, animators can control shadow casting direction intuitively.

Ground Level and Lateral Translation: To edit shadow position, animators are required to configure the ground level. This ground level is virtually set vertically onto the input animation sequence. Animators are able to adjust the level and position of the virtual ground freely so that they can reflect their preference of shadow position. As mentioned earlier, the position of shadow lets audience know where the character or object exists in a sequence. To achieve this, our approach allows animators to easily edit the ground level and position by intuitive operations.

Ground Gradient: Animators can also configure the ground gradient intuitively. This parameter enables animators to achieve shadowing such as being along the wall.

Additionally in our shadowing approach, since our input animation sequence does not have a depth, no shadow is rendered at all when the light source is placed directly above the input animation sequence (for reference, Figure 4-23 left). Therefore, we need to consider this peculiar case.

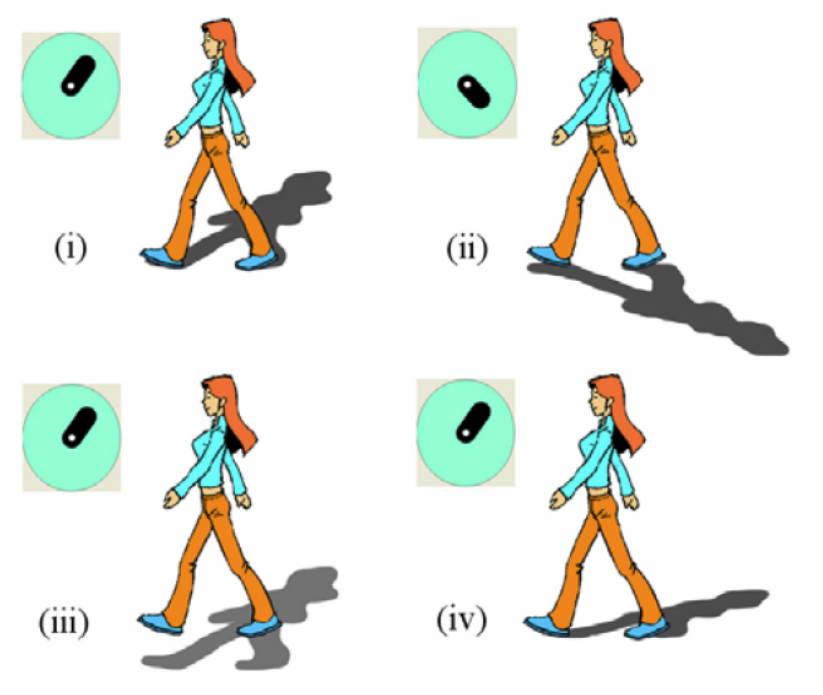

Figure 4-22, Editing Operation
(i) Initial State, (ii) Light Source Position Edit,
(iii) Ground level and Lateral Translation Edit, (iv) Ground Declination Edit,

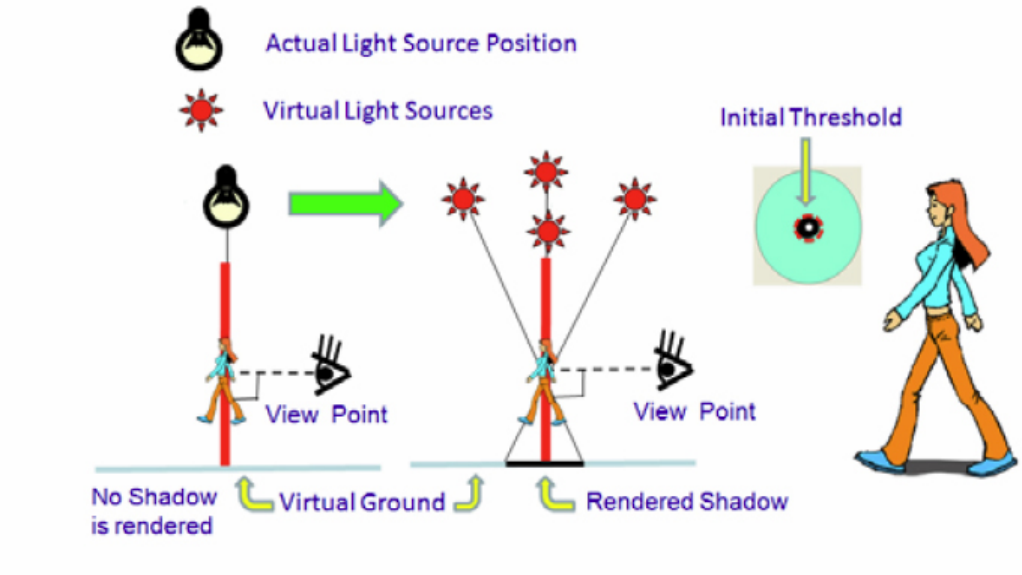

Figure 4-23, Overhead Lighting
In the case of the light source existing directly overhead of the input animation sequence (left image), the virtual light sources are set to render shadow (center image), and shadow is rendered to character's footstep (Left image)

Handling Overhead Lighting: To handle the peculiar case, we define the four virtual light sources set axisymmetrically above the input animation sequence when the light source is directly overhead above the input plane. We define this domain where the light would be set as a virtual light area by a certain threshold. Although our approach has already given the initial threshold (approximately the red dash line circle in Figure 4-16) and the number of virtual light sources, this range of the domain and even number of virtual light sources could be defined by animators. Figure 7 in the center shows an overview of the peculiar case handling algorithm in the view from the vertical direction of input animation sequence. By our approach, the shadow is rendered as shown in the right image in Figure 4-16.

Character Position Movement (Horizontally): When the actual light source is in the range of the Overhead Lighting, our approach needs to automatically control the position of virtual light sources. When the character position in the display is moving horizontally through the input animation sequence, the rendered shadow becomes misaligned gradually because the position of virtual light sources is not moving. To avoid this, we first obtain the median point of the character's alpha value in each frame. Then the distance of median point between frames is assigned to the position of virtual light sources. As a result, the rendered shadow is being under the character while the actual light source is above the input animation sequence.

Character Position Movement (Vertically): Though the character or object position in the display is moving vertically through the input animation sequence, the opacity of rendered shadow does not change at all because Shadow Map does not handle this effect. To make the animation more reality, the shadow should become sparse when the character or the object moves farther from the virtual ground. To achieve this effect, we consider a distance between the bottom of the character or object and the virtual ground. The initial shadow opacity is set to the state that the shadow is attached to the bottom of the character of object. We provide automatic opacity control for shadow based on the distance in our approach. In fact, when the character or object moves farther from the virtual ground, the shadow opacity becomes faded. By contraries, when the character or the object moves closer to the virtual ground the shadow opacity becomes dense as shown in Figure 4-17.

$$I = I_v(1-d)(0 \le I_v \le 100), (0.0 \le d \le 1.0) \tag{4.3}$$

Where I represents the opacity and d represent the distance between the bottommost of the character or object region and the virtual ground. In addition, Iv is the initial value of I. When I is set to 100, the shadow is completely opaque. On the other hand, when the value I is 0, the shadow is invisible. Our approach requires animators to set the initial value Iv first. Then the value of opacity is automatically decided depending on the value d. The origin "0.0" of d corresponds to the level of the virtual ground. And the top of the animation sequence corresponds to the value "1.0" as shown in Figure 4-24.

Although our approach can handle such opacity control, we allow animators' interaction to reflect their intension and preference to the scene. Therefore, they can choose automatic or manual opaque setting through the input animation. For example, animators can select several frames to edit the opacity from the animation sequence.

Throughout these editing and settings, the shadow shape and position is easily deformed and translated depending on animators' way of expression.

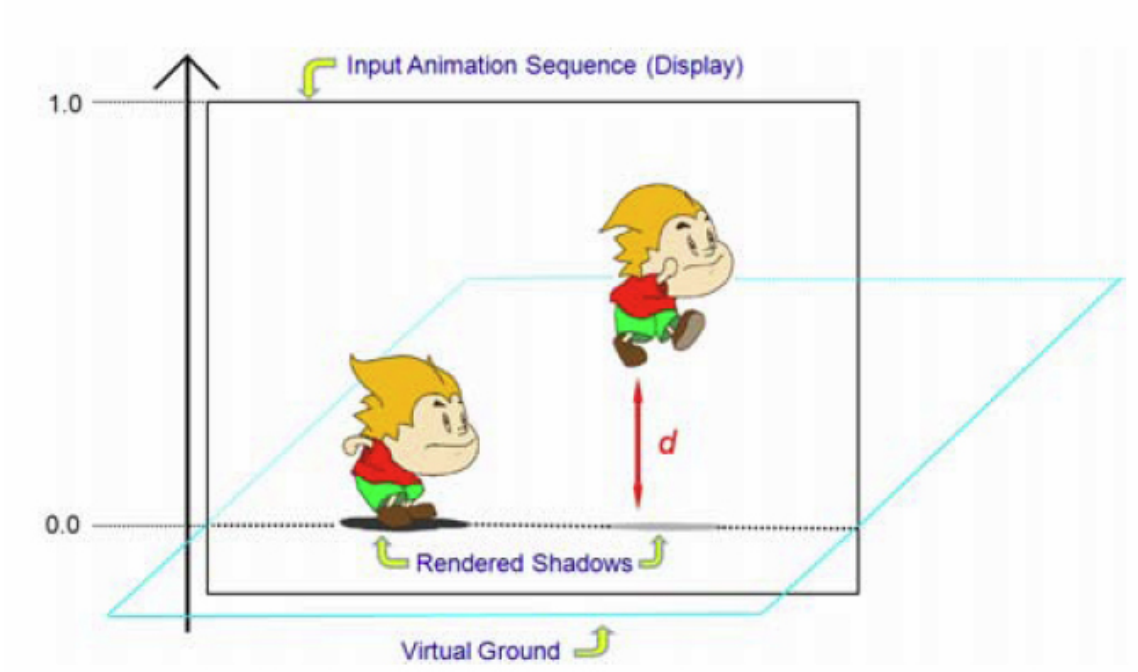

Figure 4-24, Example of Distance-Based Shadow Opacity Control: Level of virtual ground is set to the origin level (0.00) and the value at the top of the image corresponds to1.00. The value of "d" is obtained from the difference between the bottommost of the character or object region and the origin level.

4.2.33 Shadow Simplification and Blurring

Since one of our research objectives is to make shadow expression suitable for Anime, shadow simplification and blurring are considered as the key functions. To make shadows more Anime-like, we implement the shadow edge blurring and shape simplification to rendered shadow. Although several ways for blurring and simplification could be applied, we chose to apply the Gaussian Filter

$$P(X_a, Y_b) = (1/N2)\sum\sum \exp\{-(X_i - X_a)^2 / 2\sigma^2\} * \exp\{-(Y_i - Y_a)^2 / 2\sigma^2\} * P(X_i, Y_j) \quad (4.4)$$

Where, σ2 is variance, N represents the normalization constant and. Here N denotes the following equation.

$$N^2 = \sum\sum \exp\{-(X_i - X_a)^2 / 2\sigma^2\} * \exp\{-(Y_i - Y_a)^2 / 2\sigma^2\} \quad (4.7)$$

As for shadow shape simplification, the filter is applied to alpha channel of the character region. The filter blurs the border between opaque region and transparent region. As a result, the shape of character in alpha channel is gradually converted into ellipse shape. Because Shadow Map is applied to alpha channel, the rendered shadow shape also becomes ellipse shape. As for shadow blurring, in contrast, the filter is applied to the rendered shadow layer so that the rendered shadow itself is blurred.

The result of shadow shape simplification is shown in Figure 4-25(a). Only the shape of shadow is gradually simplified from left to right. In addition, the example shadow blurring is also shown in Figure 4-25(b). The shadow itself gradually blurs from left to right.

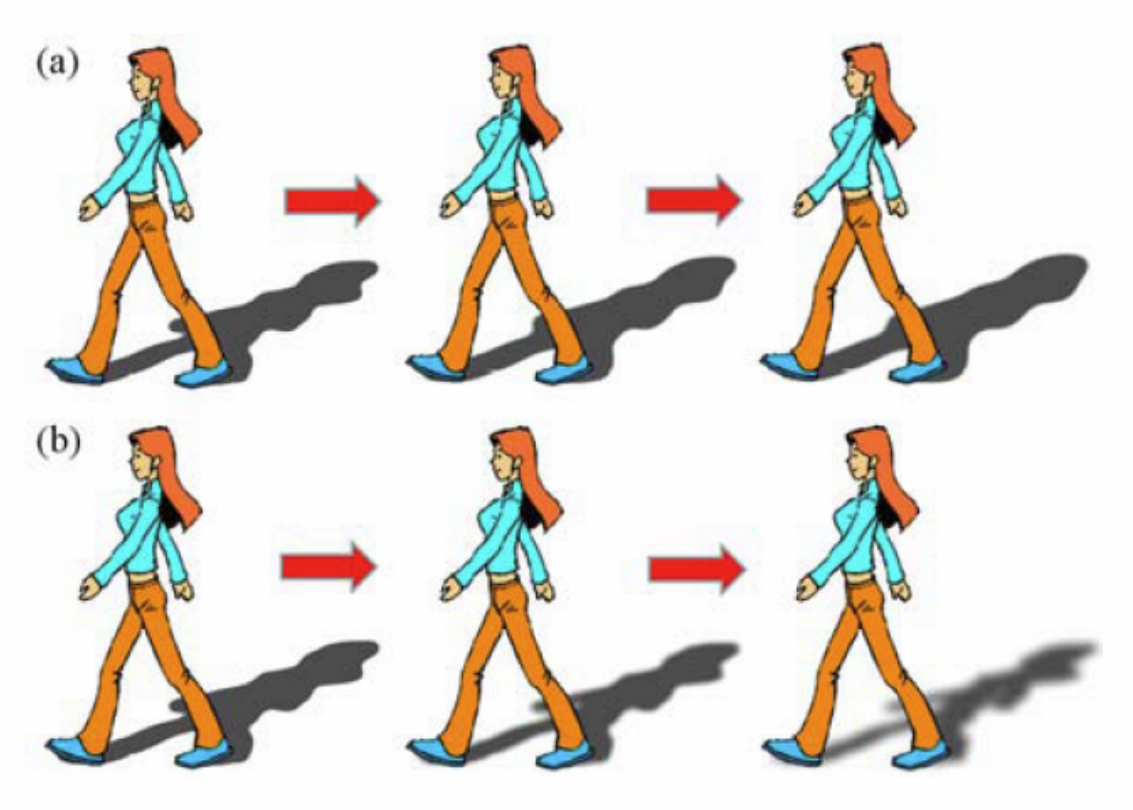

Figure 4-25, Shadow Simplification and Blurring:
(a) applying the filter to the input character region,
(b) applying the filter to the rendered shadow itself.
Although the same filter is applied, the effect is different.

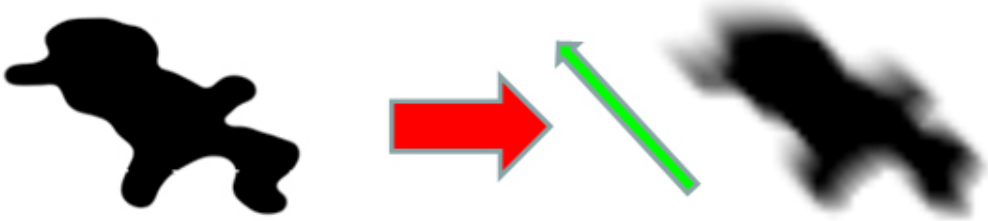

Figure 4-26, Shadow Blurring for Image Lag

4.2.34 Non-uniform Shadow Blurring

In section 4.2.32, the blurring is considered as uniform effect. Here, we introduce non-uniformed shadow blurring. This effect is considered as the image lag and can be achieved by shifting rendered shadow layer. The effect is implemented by applying the following parameters:

$$L_p * P_s * T \tag{4.6}$$

Where, Lp is direction vector which corresponds to the shadow cast direction (Green arrow in Figure 4-26). Ps represents the pixel size for shifting the shadow layer at one time. T is the number of repetition. Animators can achieve Anime-like image lag effect with adjusting these three parameters. This effect is commonly utilized in Game industry. Figure 4-26 shows the example of this effect. In this example (right image in Figure 4-26), Lp is set to the green arrow direction, Ps is set to 1.5, and the number of repetition is set to 4. This effect gives animators range of expression for shadowing.

4.2.35 2D Background Handling

Although a use of 3D background is becoming more common for the Anime productions, 2D background image is still utilized in the production. Since neither 2D background nor the input animation sequence have a depth, sometimes shadows are undesirably rendered covering the object in the background image (left image in Figure 4-28). To solve this problem, we provide a method that creates a virtual object onto the target object in the background image by using the easy image matting method [16].

As shown in Figure 4-27, the target object is easily defined by a few interactions by animators (in Figure 4-27 (b)). In fact, animators specify rough information where the target object (red lines Figure 4-27 (b)) and the background region (blue lines in Figure 4-27 (b)) are. Then the target object is extracted and a 3D plane is assigned onto the extracted object. The plane is considered as the virtual plane and exists in the same layer as the virtual ground.

Afterward, because animators can shift the plane backward and forward, the casting position of the shadow can be adjusted more appropriately. This effect makes animation more natural and attractive. The right image in Figure 4-28 is the result of virtual plane setting. The rendered shadow is successfully cast onto the surfboard.

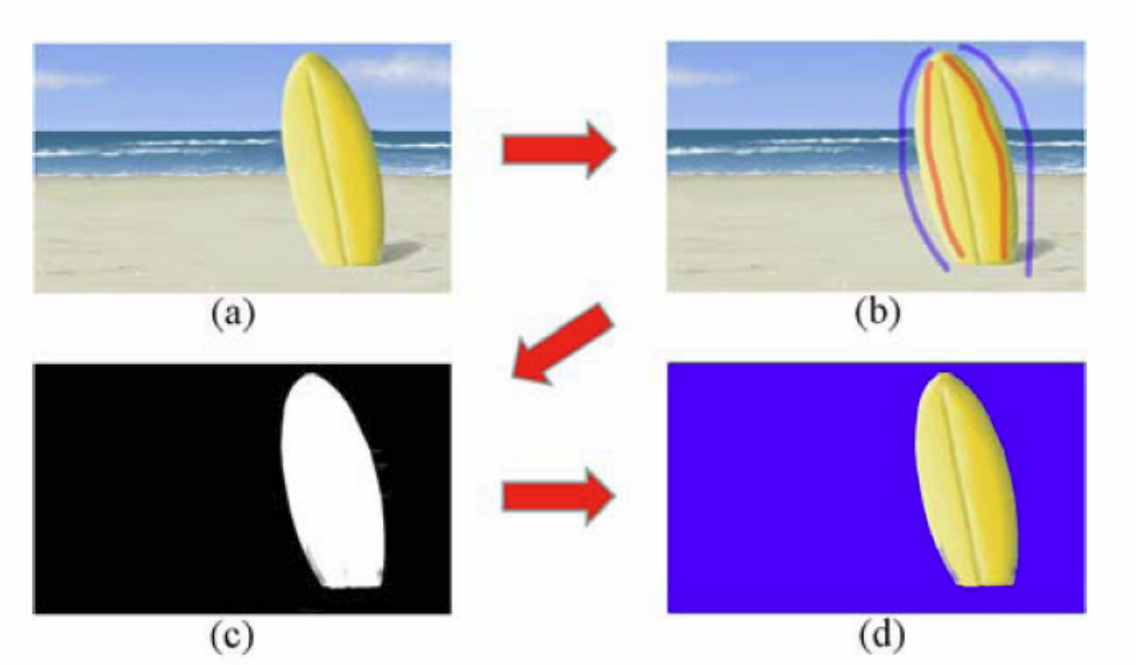

Figure 4-27, Image Matting Process:
(a) input background image, (b) animator specified strokes for masking the foreground (red curves) and background (blue curves),
(c) image matting result, (d) extracted object.

Figure 4-28, Background Handling Result:
Left: the shadow shape does not change the shape at all though the shadow is cast on the surfboard. Right: the shadow is cast on the virtual plane on the surfboard.

4.2.4 Applications for Shadowing

Our approach could be utilized in the practical Anime production.

Output Setting

The output animation sequence should be separated by each layer. In our shadowing approach, the outputs are generated as an input character or object layer, a shadow layer, and the image as seen in window. To create the final animation, those layers outputs are usually composed in commercial software such as "Adobe Premiere(TM)" or "Adobe After Effect(TM)".

Key-frame Setting

The editing parameters are influential to the entire animation sequence. This is one of the advantages of our approach. Additionally, our approach allows animators to set a key-frame shadowing so that the skilled animator can reflect their intention to the shadow rendering.

Incorporated to our CACAni system

CACAni (Computer Assisted Cel Animation) system has been developed for traditional animation production industry especially in 2D Anime [17]. The system currently is being utilized for the Anime production on a trial basis handling auto-inbetweening, auto-coloring, and layering. We have incorporated our approach into CACAni system. As a result, CACAni users also can apply our shadowing approach to their drawing (input) after auto-inbetweening and auto-coloring.

Chapter 5: Results and Discussions

The results created by our approach are shown in this Chapter and then we discuss the results.

5.1 Results and Discussions for Reference-Based Hair Motion Creation

5.1.1 Hair Animation from Reference Applied to our 3D Hair Models

From existing anime sequence, torque matching the scene is calculated by extracting hair shape to construct a hair motion database. Figure 5-1 shows that a single hair strand animation is successfully designed, proving that our approach can create a hair motion from reference hair motion.

Then, we demonstrate a hair animation to a similar model designed from our hair motion database (Figure 5-2). One advantage of our method is that we can design the hair motion to even other characters of input Anime. Figure 5-2 demonstrates our approach successfully design hair motion to another character. Since our hair motion is created based on physics model, we can design a hair motion as if two characters shared the same environment.

These results prove that we successfully animated our hair model in two animations. For each animation, we utilized the ‘on twos’ animation method, which is commonly utilized for cartoons as a means of producing more “anime-like” animation by using the same image. All animations were made at 24 frames per second, the standard rate for cartoon animation. Computations for these example results were executed on computer with a 2.4GHz Pentium 4 processor and completed within a minute for 96-frames animation for motion calculation.

5.1.2 Applications and Discussions

The other advantage of our method is that we can estimate the external force such as wind not only from hair motion, but also from the object moving by the wind in certain scene. As an example, we choose the scene that grasses move by the wind. (Figure 5-3) This would be the biggest advantage of our method. We could say that our method can create hair motions synchronously from any objects in the sequences. This result animation has also utilized the 'on-twos' method and were made at 24 frames per second.

Using hair motion data obtained from existing cartoon cel animation sequences, our method enables users to control 3D models to easily design and generate cartoon hair motions with quality close to hand-drawing cel animations. That is, hair motions peculiar to cel animation become reproducible based on the physical simulation derived from input animation sequences. Although our method uses physical equations, results animations almost synchronize input hand-drawing animations.

Also, this approach enables to apply environmental information to a three dimensional hair model such as Figure 5-3. Achieving this approach, we can put a three dimensional model character in an existing sequence as if the character exists in the existing environment from the beginning.

Since we have a three-dimensional hair model, we can handle animation with camera motion. Even though input Anime sequences are only two-dimensional, we can control three-dimensional animations (Figures 5-4 and 5-5).

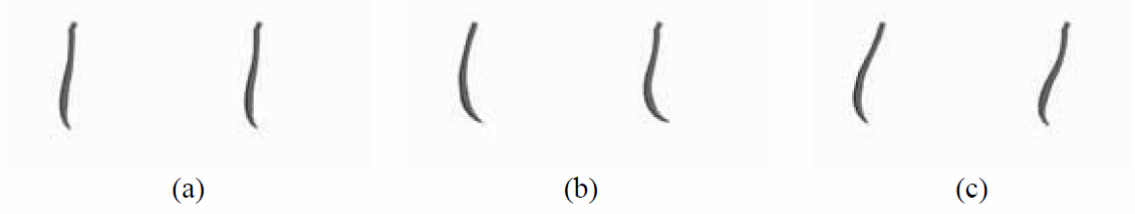

Figure 5-1, Comparing with Strand Motions
Right strand: Input sequence, Left strand: Created by our method
(a) the initial frame, (b) the 52nd frame (c) the final frame

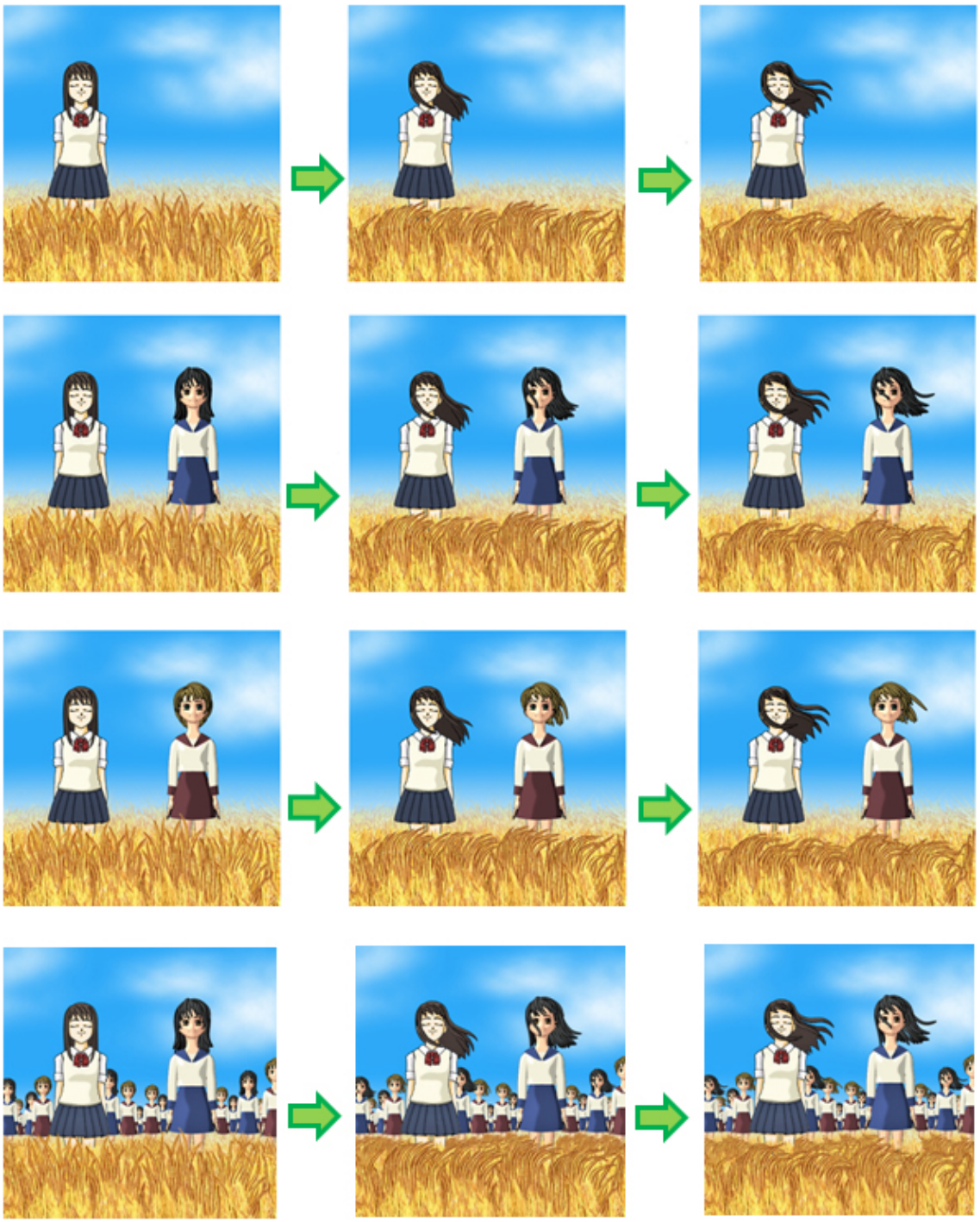

Figure 5-2, Animation Result
1st row images: Input Anime;
2nd row images: Animated by our method;
3rd row images: Our method applied to another model;
4th row: More practical example, applied to many characters

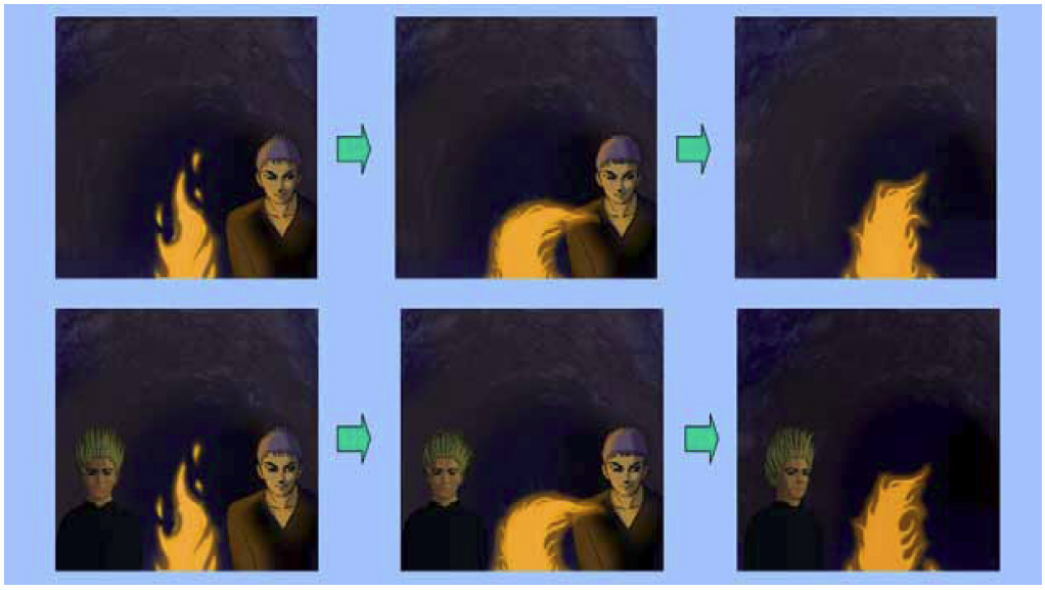

Figure 5-3, Hair Motion is applied from a fire movement.
Upper images are input sequence;
Lower images are animated by our method.

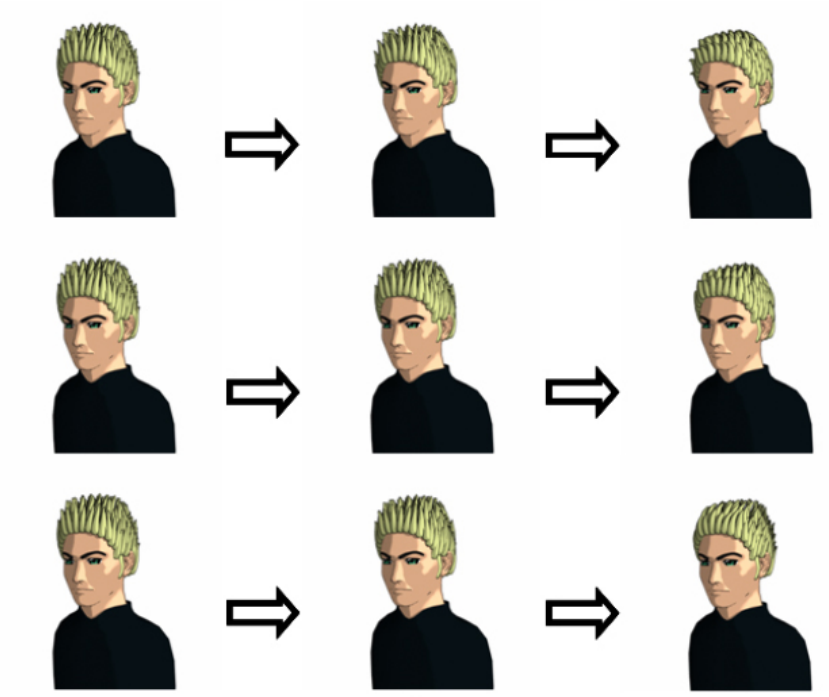

Figure 5-4, How to control z-direction force. Upper images: without z-direction force; middle images: weak z-direction force; lower images: strong z-direction force.

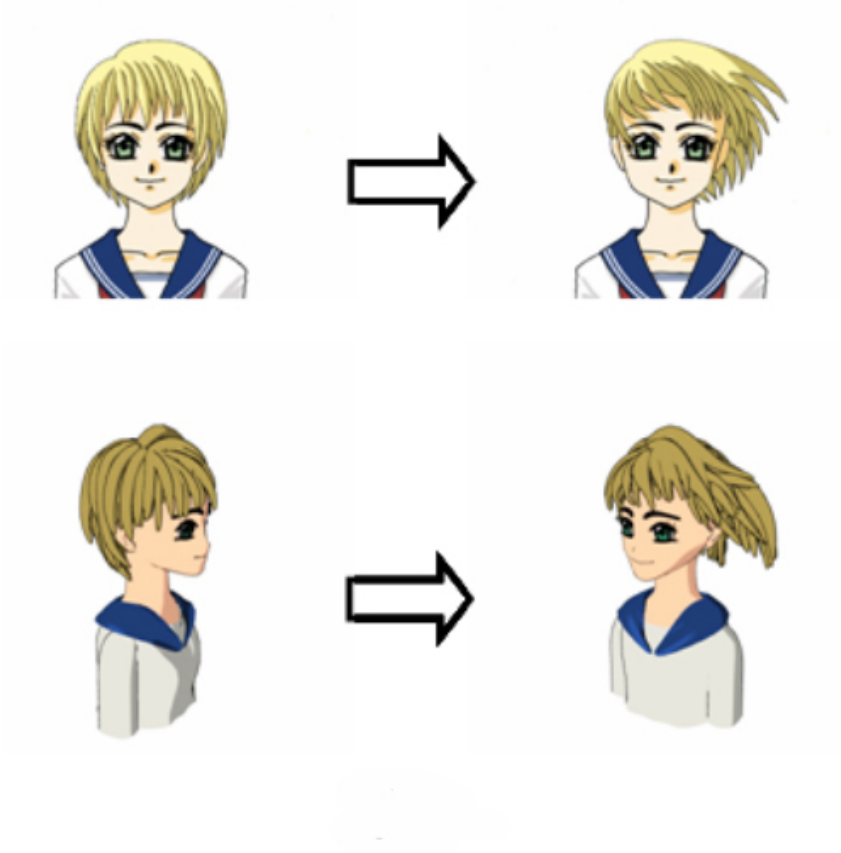

Figure 5-5, Animation with camera motion.
Top: Input Animation, Bottom: Created by the method

5.2 Results and Discussions for Simulation-Based Hair Motion Creation

5.2.1 Applied to 2D Drawings itself

Figure 5-6 shows the extracted key-frames from an obtained motion. The contour points in the first frame and the target contour points in the final frame are drawn by an animator. We suppose the wind is blowing from left to right. Since the lengths of the links in the initial frame and the ending frame are not the same, the shape of the generated contour in the ending frame does not correspond to the animator-drawn target shape. Figure 5-7 shows reconstructed key-frames with the length interpolation. In this result, the shape in the final state corresponds to animator-drawn shape. Figure 5-8 shows the generated windblown hair animation.

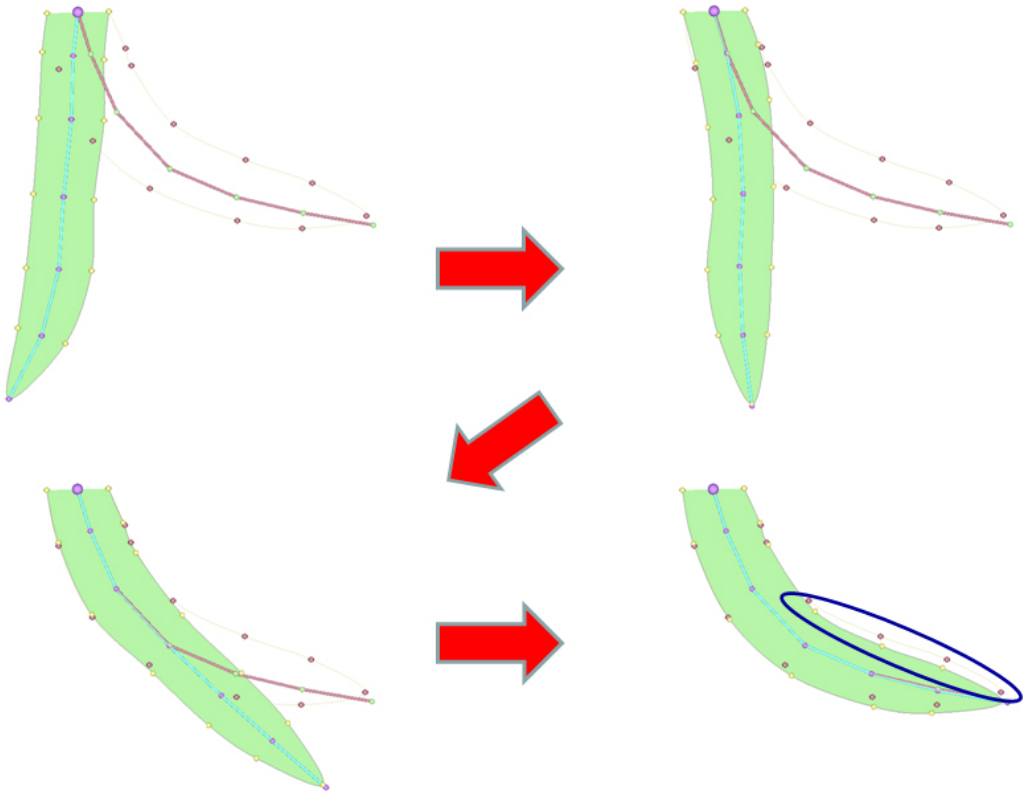

Figure 5-6, Obtained motion without Shape Reconstruction:
The hair shape does not match the target key-frame hair shape perfectly
(the bottom-right image shows the matching frame)

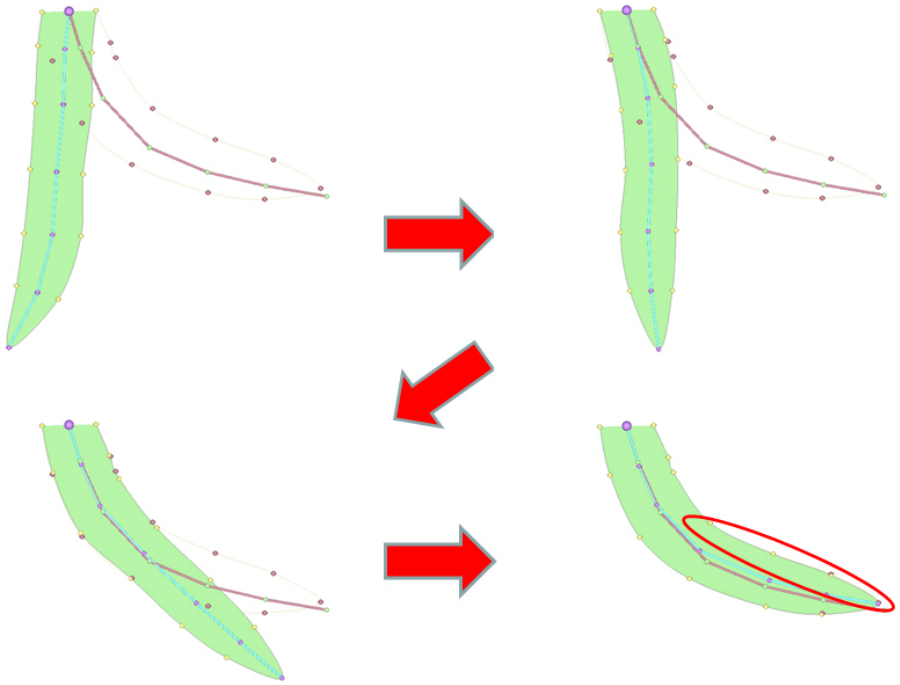

Figure 5-7, Obtained motion with Shape Reconstruction
The hair shape matches the target key-frame hair shape more faithfully
(the bottom-right image shows the matching frame)

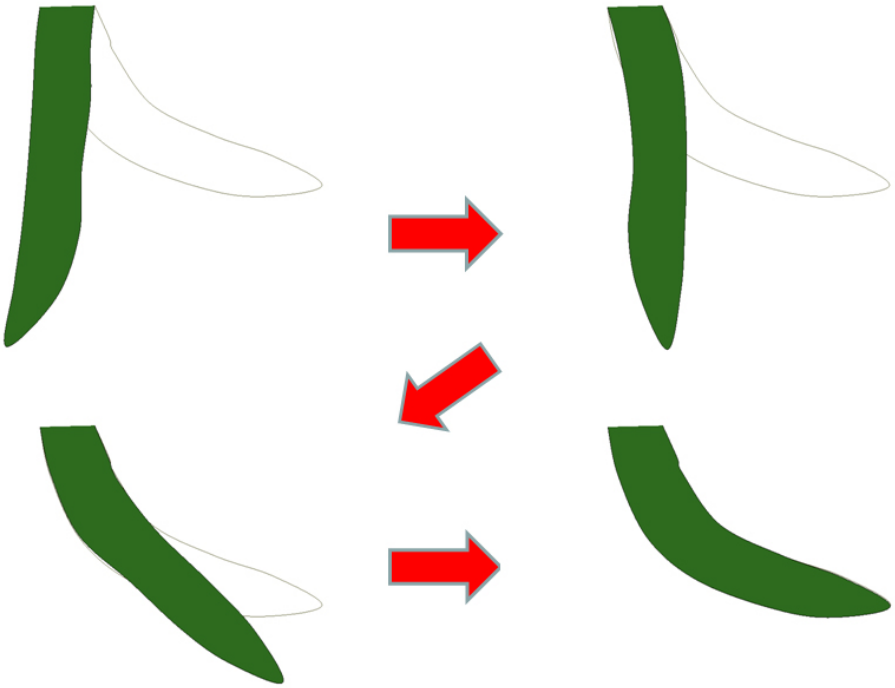

Figure 5-8, Generated Hair Strand Motion by Wind

Figure 5-9 shows the trajectory of the head movement extracted from an animation sequence drawn by an animator. For hair motion generated by the head movement, animators draw the first key-frame and input the certain positions of the head at several key-frames. The trajectory of the base is generated by interpolating these positions. The velocities and acceleration of the base are calculated. The simulation model is executed based on the motion of the base. The method creates an animation of 2 hair strands shown in Figure 5-10 on the top row. We replace these 2 hair strands with the original hand-drawn animation sequence. The original animation sequence and replaced animation sequence are shown in Figure 5-12. This result shows our method can successfully create in-between frames for hair animation which is similar to hand-drawn ones.

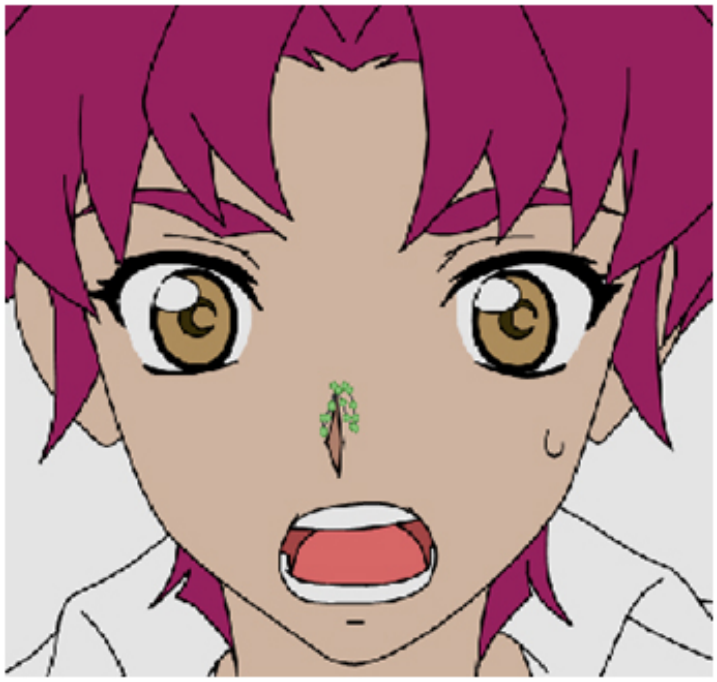

Figure 5-9, Trajectory of the Head Movement
(Copyright by Anime International Co.)

Figure 5-10, Comparison between Hand-Draw Animation (left column) and simulated animation (right column) of several hair strands (Copyright by Anime International Co.)

5.3 Results and Discussions for Interactive and Intuitive Shadowing

We have proposed a novel approach on creating character animation with shadow. The animation (Figure 5-11, 5-12) is created on an Intel Xeon 2.0 GHz computer with NVIDIA Quadro FX1700, and DirectX 9.0 as the graphics API. The input size is 1000 * 600 pixels which could be utilized as a previewing in the industry. As for editing shadows, it is practically in real-time for all operations. Figure 5-11 demonstrates a sequence of shadow animation. The upper row is the input image sequence, the second row is the animation without shadow, the third row is the animation with relatively-detailed shadow, and the bottom row is the animation with Anime-Like shadow. Usually in Anime, detailed shadow is not always required because too much detail of the shape may distract the scene. Therefore the shadow tends to be rendered such as the bottom row images. Moreover our approach enables animators not only to create and edit shadow in the first frame, but also in several key-frames independently from the inbetweening for input animation sequence. Figure 5-12 demonstrates another sequence of shadow animation. The upper row is the input image sequence, the middle row is the animation without shadow, and the bottom row is the animation with Anime-Like.

There are two advantages of using the Shadow Map for our approach to be utilized in production pipeline. One is the high-speed performance that enables interactive and intuitive shadow editing in real-time.

The other advantage is that the Shadow Map is flexible with various backgrounds. Obviously shadows can be rendered on a flat ground. Furthermore, our approach can handle the 2D background image. In fact it could also be utilized for complicated backgrounds such as a 3D background. When the shadow map is applied to the input image sequence in a 3D environment, because the background is already 3D, there need not be additional processing involved unlike the 2D background handling we described previously.

Although simple affine transformation to the character or object region, calculated by the relationship between the input animation sequence, light, and ground, is possible for casting shadows on the flat ground it is not practical for such a complicated background.

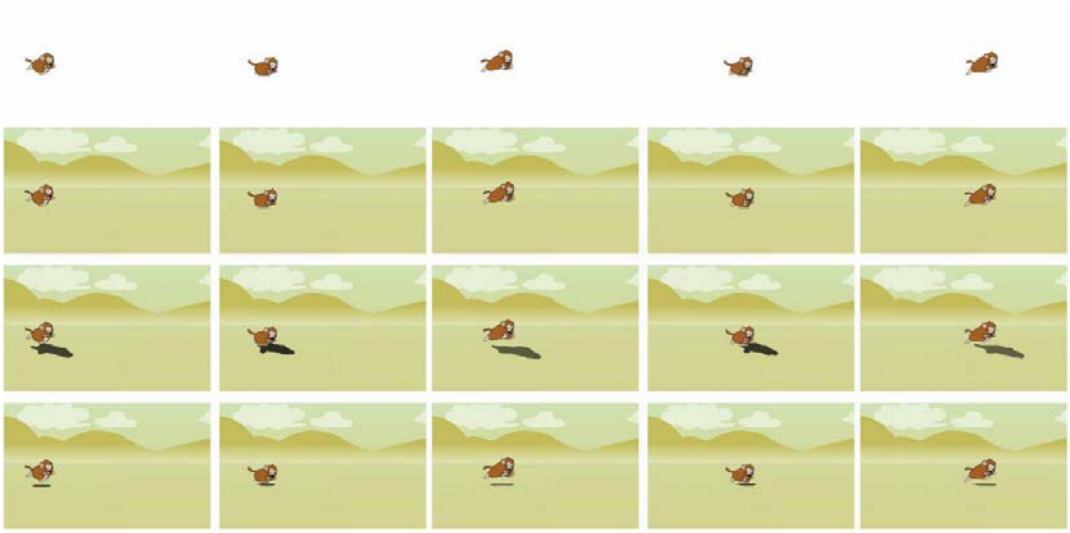

Figure 5-11, Shadow Animation Result I (Monkey Run): Only in the first frame is edited by animator. Top row: input animation sequence. The second row: animation without shadows. The third row: animation with relatively-detailed shadow. The bottom row: animation with Anime-like shadow.

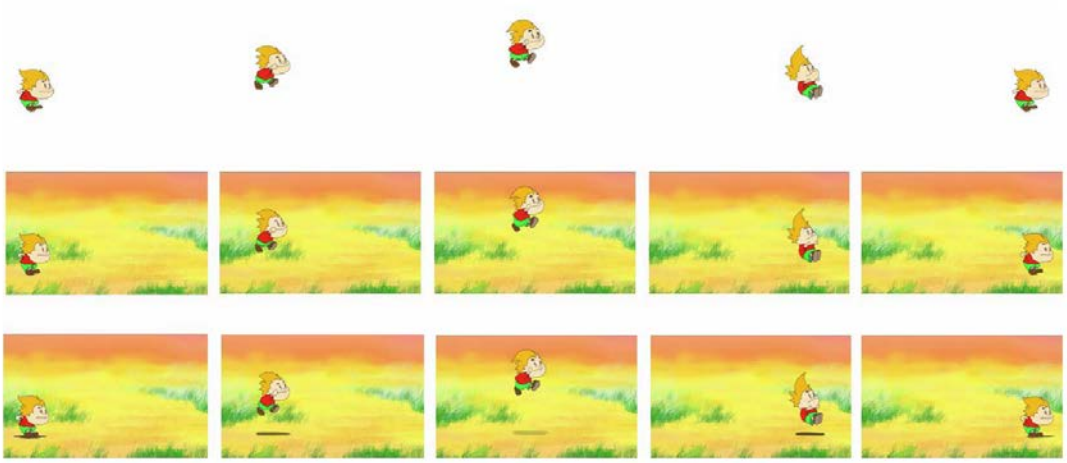

Figure 5-12, Shadow Softness Handling
Shadow position automatically follows the character.
Top row: input animation sequence.
The middle row: animation without shadows.
The bottom row: animation with Anime-like shadow.

Chapter 6: Conclusion and Future Work

As for the Simulation-Based Hair Motion Creation since the target Anime-Like expression exists in the animator`s mind, the animation results could be considered to be the right expression by one animator but not by another animator. An extreme way of saying this is that an Anime character has a life in all scenes. What the character does must have the meaning that an animator wants to express. One can argue that this is why various methods of cartoon expression have been developed.

Allowing this inconsistency is a huge advantage of cartoon expression and the most difficult thing to express in computer graphics. Even if the appearance of a cartoon characters` hair is physically impossible, the animation can still be fantastic and express the animator's intention. This is an advantage of drawing by hand. Since the model utilizes a 3D hair structure to render hair, it cannot handle such inconsistencies. The question of how to address these inconsistencies is future work. Automating the pre-processing steps from the hand-drawn input is also left for future work.

In terms of the Reference-Based Hair Motion Creation, of course, there are a few limitations in this approach. Since this approach needs to construct physical equations from the difference of angle between existing frames, input data cannot be created from sequence with camera motion. Also, this method may not handle if the difference of angle is too large. Sometimes in cartoon, there are "super-exaggerate" expressions. In this case, our approach is occasionally collapse.

One of the purposes of this research is to achieve the motion from animators' taste. In terms of hair motion, this method has accomplished attractive motions. In terms of discrepancy of the hair number, however, we need to effort more to achieve this kind of expression. Allowing an inconsistency we have mentioned is a huge advantage of cartoon expression. At the same time, one of the most difficult things to express in three dimensional computer graphics. Even if the appearance of cartoon character's hair is physically impossible, the animation can still be fantastic and express the animators' intention. This is an advantage of hand-drawing. Since the model utilizes a three dimensional hair structure to render hair, it is difficult to deal with such inconsistencies. The question of how to address these inconsistencies goes to future work. We consider that this boils down to rendering problem.

Future projects are consideration of how shadowing, rendering, etc. should be handled to enable discrepancy in cartoon expression. In addition, the future work about input,

we can utilize hair motion capture data as reference data. Since motion capture data has information for whole sequence, we do not have to create inbetween frames. Instead of creating in-between frames, we need to choose feature frames from the input sequence, which has Anime-like feature. How we set the selection function is going to be future work.

With regard to Interactive and Intuitive Shadowing, several issues need to be addressed in our future work. Although our approach allows animators to apply 2D background, we currently can handle simple background image. Our solution cannot handle the more complicated design of 2D background such as grass field, or a field that the shadow is already drawn on. When the rendered shadow covers the grass and/or the pre-drawn shadow, the shape or color of the shadow should be changed accordingly. A more flexible and interactive approach for the background handling is to be implemented. Furthermore, more intuitive and interactive shadow editing such as local shape deformation, boundary emphasis, and multi-light source handling such as [49] should be incorporated.

References

[1] Madeira J S, Stork A, and Grob M H. An approach to computer-supported cartooning. *Visual Computer*, (12:1C17), 1996.

[2] Patterson J W and Willis P J. Computer assisted animation: 2D or not 2D? *The Computer Journal*, (37(10)):829-839, 1994.

[3] Durand C X. The TOON project: requirement for a computerized 2D animation system. In *Proceedings of ACM SIGGRAPH*, pages 285-295, 1991.

[4] Thalmann N M and Thalmann D. Computer animation: theory and practice. Springer-Verlag, 1985.

[5] Halas J and Manvell R. *The technique of ¯lm animation*. Hastings House, New York, 1968.

[6] Fekete JD, Bizouarn E, Cournarie E, Galas T, and Taillefer F. TicTacToon: a paperless system for professional 2D animation. In Computer Graphics Proceedings, Annual Conference Series, pages 79-90, 1995.

[7] Burtnyk N and Wein M. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. In Communications of the ACM 19, pages 564-569, 1976.

[8] *http://www.cambridgeanimation.com/products/animo.htm/.*

[9] Sederberg T W, Gao P, Wang G, and Mu H. 2-D shape blending: an intrinsic solution to the vertex path problem. In Computer Graphics Proceedings, Annual Conference Series, pages 15-18, 1993.

[10] *http://www.toonboom.com/products/.*

[11] *http://www.retas.com/.*

[12] *http://www.macromedia.com/software/flash/*.

[13] *http://www.celaction.com/*.

[14] *http://www.bauhaussoftware.com/products mirage studio.php*.

[15] Ken. Anjyo, Y. Usami, and T.Kurihara. A simple method for extracting the natural beauty of hair. In Proc. of SIGGRAPH 92, pp. 111 - 120, 1992.

[16] R.E. Rosenblum, W.E. Carlson, and E. Tripp. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. The Journal of Visualization and Computer Animation, pp.141-148,

[17] Tae-Yong Kim and Ulrich Neumann "Interactive Multiresolution Hair Modeling and Editing", In Proceedings of SIGGRAPH 2002 San Antonio pp 620 - 629.

[18] Y.Yu. Modeling realistic virtual hairstyles. In Proceedings of Pacific Graphics, pp. 295-304, 2001.

[19] Hadap and N. MagnenatThalmann. Interactive hair styler based on fluid flow. In Computer Animation and Simulation 2000. Proceedings of the Eleventh Eurographics Workshop, 2000.

[20] J. T. Chang, J. Jin, and Y. Yu."A practical model for hair mutual interactions" Proceedings of ACM SIGGRAPH Symposium on Computer Animation 2002, 73 - 80, 2002.

[21] E. Plante, M.-P. Cani, and P. Poulin. A layered wisps model for simulating interactions inside long hair. In Proceedings of Eurographics Computer Animation and Simulation, 2001.

[22] Y. Bando, B.-Y. Chen and T. Nishita, Animating Hair with Loosely Connected Particles, Computer Graphics Forum, Vol. 22, No. 3, 2003.

[23] John Lasseter "Principle of Traditional Animation Applied to 3D Computer Animation" Proc Siggraph 1987, pp 35-44.

[24] P. Noble and W. Tang: Modelling and Animating Cartoon Hair with NURBS Surfaces, Proc. CG International 2004, pp. 60 - 67.

[25] Paul Rademacher, “View-Dependent Geometry” In Proceedings of SIGGRAPH 99, L.A pp. 439-446

[26] Y.Wei, E.Ofek L.Quan and H-Y.Shum, “Modeling Hair from Multiple Views”, In Proceedings of ACM SIGGRAPH, pp816 - 820 2005.

[27] K.Zhou, J.Huang, J.Anyder, X.Liu, H.Bao, B.Guo, and HY. Shum, “Large Mesh Deformation Using the Volumetric Graph Laplacian”, In Proceedings of ACM SIGGRAPH, pp496 - 503 2005.

[28] Williams H L, Casting curved shadows on curved surfaces. In Proceedings of SIGGRAPH ’78, pages 270-274, 1978.

[29] Segal. M, Korobkin C, Widenfelt R Foran J, Haeberli P, Fast shadows and lighting effects using texture mapping. In Proceedings of SIGGRAPH ’92, pages 249-252, 1992.

[30] Staimminger, M., and Drettakis, G., Perspective shadow maps. ACM Transaction on Graphics, 21, 3, 2002. Pages 557-562.

[31] Wimmer M, Scherzer D and Purgathofer W, Light Space Perspective Shadow Maps, In Proceedings of Eurographics Symposium on Rendering 2004

[32] Martin T, Tan T-S Anti-aliasing and Continuity with Trapezoidal Shadow Maps, In Proceedings of Eurographics Symposium on Rendering 2004 pages 153-160.

[33] Lloyd B., Yoon S., Tuft D., and Manocha D. Subdivided Shadow Maps. Technical Report TR05-024, University of North Carolina at Chapel Hill, 2005.

[34] Lloyd B., Govindaraju N., Quammen C., Molnar S., and Lloyd B. Logarithmic Perspective Shadow Maps. Technical Report TR07- 005, University of North Carolina at Chapel Hill, June 2007.

[35] Donnelly W. and Lauritzen A. Variance shadow maps. In I3D '06: In Proceedings of the 2006 symposium on Interactive 3D graphics and games 2006, pages 161-165, ACM Press.

[36] Nakajima, H., Sugisaki, E., Welmler. S., and Morishima, S. Automatic Shadowing from Anime Sequence Layers. Poster & Animation Proceedings of the Symposium on Non-Photorealistic Animation and Rendering, 5, 2008.

[37] Petrovic, L., Fujito, B., Williams, L., and Finkelstein, A. Shadows for cel animation. In Proceedings of SIGGRAPH2000, pages 511- 516.

[38] Pellacine, F., Tole, P., and Greenberg, D. 2002. A user interface for interactive cinematic shadow design. ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH2002 21, 3, 563- 566.

[39] Decoro, C., Cole, F., Finkekstein, A., and Rusinkiewicz, S. Stylized Shadows. Proceedings of the 5th international symposium on Nonphotorealistic animation and rendering, pages 77-83.

[40] Nakajima H., Susgisaki E., and Morishima S. Tweakable Shadows for Cartoon Animation. In Proc. of the 15th international Conference in Central Europe on Computer Graphics 2007, pages 233-240.

[41] http://www.kuffner.org/james/software/index.html,
Multi-body Dynamics (Package software)

[42] P. Litwinowicz, L. Williams. Animating images with drawings. SIGGRAPH 94, Orlando, FL, pp. 409 - 412,1994

[43] Lucas Kovar, Michael Gleicher , and Fred Pighin, Motion Graph, In Proceedings of SIGGRAPH 2002 San Antonio. pp 473-482.

[44] J. P. Lewis, Matt Cordner, Nickson Fong,”Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation” In Proceedings of SIGGRAPH 2000 New Orleans. pp 165 - 172.

[45] Doug L. James and Kayvon Fatahalian, Precomputing Interactive Dynamic Deformable Scenes, In Proceedings of ACM SIGGRAPH, 2003.

[46] Advanced RenderMan, A.A.Apodaca and L. Gritz. Morgan Kaufmann 2002

[47] Lokovic, T., and Veach, E. Deep shadow maps. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques 2000, pages 385-392.

[48] Annen T., Dong Z., Mertens T., · Bekaert P., Seidel P Kautz J., Real-Time, All-Frequency Shadows in Dynamic Scenes ACM SIGGRAPH 2008 TRANSACTIONS ON GRAPHICS, Volume 27 , Issue 3 pp.

[49] Wang, J., Drucker, M. S., Agrawala M., Cohen F. M., The Cartoon Animation Filter, ACM SIGGRAPH 2006 Transaction on Graphics Volume 25, Issue 3, 2006.7

[50] Johnston, O., and Thomas, F. 1995. The Illusion of Life: Disney Animation, Disney Editions.

[51] Johnston SF. Lumo: Illumination for cel animation. In Proceedings of NPAR 2002, pages 45–52 and 156, 2002.

[52] Shimotori Y, Nakajima H, Sugisaki E, Maejima A, and Morishima S. Interactive shade control for cartoon animation. In ACM SIGGRAPH 2007 posters, page ArticleNo. 170, 2007.

[53] Anjyo K and Hiramitsu K. Stylized highlights for cartoon rendering and animation. IEEE Computer Graphics and Applications, 23(4):54–61, 2003.

[54] Anjyo K, Wemler S, and Baxter W. Tweakable light and shade for cartoon animation. In Proceedings of NPAR 2006, pages 133–139, 2006.

[55] Todo H, Anjyo K, Baxter W, and Igarashi T. Locally controllable stylized shading. ACM Transactions on Graphics (Siggraph 2007), 26(3):Article No. 17, August 2007.

[56] Hiroshi YASUDA, Ryota KAIHARA, Suguru SAITO and Masayuki NAKAJIMA, "Motion Belts: Visualization of Human Motion Data on a Timeline", IEICE Transactions on Information and Systems 2008 E91-D(4):1159-1167

[57] “Image Processing Based on Partial Differential Equations” Tai, X.-C. et al. ed. Springer-Verlag 2007. 440 pp. (H) ISBN 3-540-33266-9

[58] M. Lysaker, S. Osher, and X.-C. Tai. Noise removal using smoothed normals and surface fitting. IEEE Trans. Image Processing, 13(10):1345–1457, 2004.

[59] R. Featherstone. Robot Dynamics Algorithms. Kluwer Academic Publishers, 1987.

[60] Multi-body Dynamics (Package software)
http://www.kuffner.org/james/software/index.html

[61] BAI X., LATECKI L. J.: Discrete skeleton evolution. In Proceedings of the 6th Int. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR 2007), LNCS4679 (2007), pp. 362–374.

[62] BAI X., LATECKI L. J., LIU W.: Skeleton pruning by contour partitioning with discrete curve evolution. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 3 (2007), 449–462.

[63] BAILEY D. G.: An efficient Euclidean distance transform. In Proceedings of the International Workshop on Combinatorial Image Analysis, LNCS 3322 (2004), pp. 384–408.

[64] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As rigid-as-possible shape manipulation. ACM Transactions on Computer Graphics 24, 3 (2005), 1134–1141

[65] WANG C., WANG Z., ZHOU Q., SONG C., GUAN Y., PENG Q.: Dynamic modeling and rendering of grass wagging in wind. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 377–389.

[66] FARRIS B., EL-SAWI K.: Physics-based robotic simulation using joint constraint. In *Proceedings of the 13th Annual Conference on Industry and Management Systems* (2007), pp. 53–58.

[67] FEATHERSTONE R.: The calculation of robot dynamics using articulated body inertias. *International Journal of Robotics Research 2*, 1 (1983), 13–30.

[68] ANJYO K., ARIAS M., HORRY Y., MOMOSE Y.: Digital cel animation in Japan. In Siggraph 2000 Conf. Abstracts and Applications (2000), ACM Press, pp. 115–117.

[69] KRIKKE J.: Computer graphics advances the art of anime. IEEE Computer Graphics and Applications 26, 3 (2006), 14–19.

[70] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (Siggraph 1999) (1999), pp. 409–416.

[71] KARPENKO O. A., HUGHES J. F.: Smooth Sketch: 3D free-form shapes from complex sketches. In Proceedings of the 33th annual conference on Computer graphics and interactive techniques (Siggraph 2006) (2006), pp. 589–598.

[72] MASNOU S., MOREL J.-M.: Level lines based disocclusion. In Proc. IEEE Int. Conf. on Image Processing, Chicago, IL (1998), pp. 259–263.

[73] BERTALMIO M., SAPIRO G., CASELLES V., BALLESTER C.: Image inpainting. In Computer Graphics, SIGGRAPH 2000 (2000), pp. 417–424.

[74] BERTALMIO M., BERTOZZI A. L., SAPIRO G.: Navier-Stokes, fluid dynamics, and image and video inpainting. In Proc. Conf. Comp. Vision Pattern Rec. (2001), pp. 355–362.

[75] TAI X.-C., OSHER S., HOLM R.: Image inpainting using TV-Stokes equation. In Image Processing Based on Partial Differential Equations (2006), Springer, Heidelberg, pp. 3–22.

[76] BALLESTER C., BERTALMIO M., CASELLES V., SAPIRO G., VERDERA J.: Filling-in by joint interpolation of vector fields and gray levels. IEEE Trans. Image Processing 10, 8 (2001), 1200–1211.

[77] CHAN T. F., KANG S.-H., SHEN J.: Euler's elastic and curvature based inpaintings. SIAM J. Appl. Math.. 63, 2 (2002), 564–594.

[78] CHAN T. F., SHEN J.: Variation image inpainting. Comm. Pure Applied Math. 58 (2005), 579–619.

[79] WU T.-P., TANG C.-K., BROWN M., SHUM H.-Y.: Shape palettes: Interactive normal transfer visa sketching. ACM Transactions on Graphics volume 26 Issue 3 07 (2007), Article No.44.

[80] WU T.-P., SUN S., TANG C.-K., SHUM H.-Y.: Interactive normal reconstruction from a sigle image. ACM Transactions on Graphics volume 27 Issue 5 08 (2008), Article No.119.

[81] NG H. S., WU T.-P., TANG C.-K.: Surface-from gradients with incomplete data for single view modeling. In ICCV07 (2007), pp. 1–8.

[82] BARLA P., THOLLOT J., MARKOSIAN L.: X-toon: An extended toon shader. In International Symposium on Non-Photorealistic Animation and Rendering (NPAR) (2006), ACM.

[83] ASSA J., CASPI Y., COHEN-OR D.: Action synopsis: pose selection and illustration. ACM Transactions on Computer Graphics 24, 3 (2005), 667–676.

[84] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? In Proceedings of the 35th annual conference on Computer graphics and interactive techniques Siggraph 2008.

[85] FRANKOT R. T., CHELLAPPA R.: A method for enforcing integrability in shape from shading algorithms. IEEE Trans. Pattern Anal. Mach. Intell. 10, 4 (1988), 439–451.

[86] PETROVIC N., COHEN I., FREY B. J., KOETTER R., HUANG T. S.: Enforcing integrability for surface reconstruction algorithms using belief propagation in graphical models. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 1 (2001), 743–748.

[87] WU T.-P., TANG C.-K.: Visible surface reconstruction from normals with discontinuity consideration. In CVPR ’06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Washington, DC, USA, 2006), IEEE Computer Society, pp. 1793–1800.

[88] TAI X.-C., WU C.: Augmented lagrangian method, dual methods and split bregman iteration for rof model. In SSVM ’09: Proceedings of the Second International Conference on Scale Space and Variational Methods in Computer Vision (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 502–513.

[89] CHEN Q., WU Z., TIAN F., SEAH H. S., QIU J., MELIKHOV K.: Vectorization of raster line-drawings in cartoons. In Proceedings of International Conference on Computer Animation and Social Agents, CASA (Oct. 2005), pp. 151–158.

[90] Chen, Q. 2008. Computer-Assisted Inbetween Generation for Cel Animation. PhD thesis, Nanyang Technological University.

[91] 東京工科大学 デジタルアニメ政策技術研究会監修, “デジタルアニメマニュアル2008”.

Publication

1. “Interactive Shadowing for 2D Anime”, A Journal of Computer Animation and Virtual Worlds, Volume 20. PP.395 - 404, 2009.6, Eiji Sugisaki, Hock Soon Seah, Feng Tian, Shigeo Morishima.

2. “直感的に影を演出可能な編集ツール”，映像情報メディア学会誌, Vol.62, No.2, pp.234-239, 2008.2, 中嶋 英仁, 杉崎 英嗣, 森島 繁生,

3. “モーションキャプチャシステムを用いた頭髪アニメーション手法の提案”, 画像電子学会誌 ビジュアルコンピューティング論文特集号, vol.36, No.4, pp.398-406, 2007.7 杉崎 英嗣, 風間 祥介, 石川 貴仁, 白石 允梓, 西村 昌平, 森島 繁生,

4. “Hair Motion Cloning from Cartoon Animation Sequences”, A Journal of Computer Anima-tion & Virtual World July 2006, vol.17, pp.491-499, 2006.7. Eiji Sugisaki, Yosuke Kazama, Shigeo Morishima, Natsuko Tanaka, Akiko Sato

5. “In-Between Creation for Anime Character’s Hair”, will be published in Proceeding of International Workshop on Advanced Image Technology 2010, 5-page paper 2010.1, Eiji Sugisaki, Fumihito Kyota, Hock Soon Seah, Masayuki Nakajima

6. “Simulation-Based In-Between Creation for CACAni System”, will be published in SIGGRAPH ASIA 2009 Sketches & Posters. 2009.12, Eiji Sugisaki, Fumihito Kyota, Hock Soon Seah, Masayuki Nakajima

7. “ Instant and Intuitive Shadowing for CACAni System”, Proceeding of International Workshop on Advanced Image Technology 2009, Article No.92 6-page paper 2009.1, Eiji Sugisaki, Feng Tian, Hock Soon Seah

8. “手描きアニメを利用した影生成システム ～影造～”，第7回NICOGRAPH 春季大会 論文&アート部門コンテスト論文集, pp.72-77, 2008.3, 中嶋 英仁, 杉崎 英嗣, 上村 周平, 森島 繁生,

9. “Directable and Stylized Hair Simulation”, INSTICC GRAPP2008 Conference Proceedings DVD-ROM, pp.316-321, 2008.1, Yosuke Kazama, Eiji Sugisaki, Shigeo Morishima,

10. "影を自在に演出可能な影編集ツール", 第3回デジタルコンテンツシンポジウム, p1-4.pdf, 2007.6, 中嶋 英仁, 杉崎 英嗣, 森島 繁生,

11. “ Tweakable Shadows for Cartoon Animation” In WSCG FULL Papers proceedings, ISBN 978-80-86943-01-5, pp.233-240, 2007.2, Hidehito Nakajima, Eiji Sugisaki, Shigeo Morishima,

12. “Anime Hair Motion Design from Animation Database”, CyberGames 2006/The Joint Inter-national Conference on CyberGames and Interactive Entertainment 2006 (CGIE2006), ISBN: 86905-901-7, SS1-4.pdf, 2006.12, Eiji Sugisaki, Yosuke Kazama, Shigeo Morishima, Natsuko Tanaka, Akiko Sato,

13. “Simulation-Based Cartoon Hair Animation”, The 13-th International Conference in Central Europe on Computer Graphics,pp.117-122,2005.2, Eiji Sugisaki, Yizhou Yu, Ken Anjyo, Shigeo Morishima,